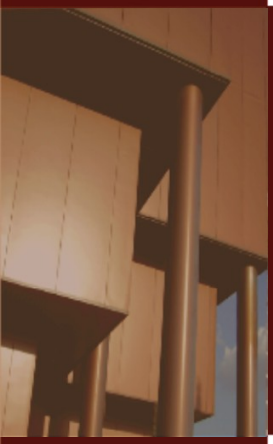
Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Estudio del uso de Redes Neuronales Artificiales como herramienta para detectar el cáncer de ano

Autor: Manuel Lazo Maestre

Tutoras: Carmen Serrano Gotarredona, Begoña Acha Piñero y Cristina Suárez Mejías



Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Estudio del uso de Redes Neuronales Artificiales como herramienta para detectar el cáncer de ano

Autor:

Manuel Lazo Maestre

Tutoras:

Carmen Serrano Gotarredona

Begoña Acha Piñero

Cristina Suárez Mejías

Dpto. de Teoría de la Señal y Comunicaciones

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2019

Proyecto Fin de Carrera: Formato de Publicación de la Escuela Técnica Superior de Ingeniería de Sevilla

Autor: Manuel Lazo Maestre

Tutoras: Carmen Serrano Gotarredona, Begoña Acha Piñero y Cristina Suárez Mejías

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Querría aprovechar esta página para agradecer de forma general a todo aquel que ha nutrido mi instrucción y educación. Es necesario aprender a aprovechar todo el conocimiento al que nos exponemos, en cualquier entorno, no limitarnos al académico. Es a veces en aquellas situaciones mas cotidianas donde ha de filtrarse con un tamiz enormemente fino para rielar aquellos detalles imperceptibles que apoyan los conceptos de mayor fuerza. Si como decía Picasso: "que la inspiración te llegue trabajando", podríamos decir también: "que el conocimiento te llegue pensando".

M.L.

Sevilla 2019

Resumen

Este trabajo trata de poner las bases de un sistema de ayuda al diagnóstico que los médicos puedan utilizar a la hora de tratar el cáncer de ano. Para ello se basa en el procesamiento de imágenes clínicas y en el entrenamiento de redes neuronales.

En términos mas generales, se desea contrastar la eficacia de un método como el de la inteligencia artificial para el tratamiento de la enfermedad, porque no existen casi soluciones de apoyo a la decisión que faciliten el trabajo de los médicos.

Abstract

This work tries to lay the foundation of an aided diagnosis system for the treatment of anal cancer. It is based on the image preprocessing and artificial neural network's training.

It's intention is to understand the viability of an artificial inteligence aproach to diagnose the disease. Because there is almost no other investigation on the subject, the main interest is to help doctors in the clinical environment.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE BLOQUES DE CÓDIGO

Notación

HRA	High Resolution Anoscopy
HSH	Hombres que tienen Sexo con Hombres
HPV	Human Papilloma Virus
VIH	Virus de Inmunodeficiencia Humana
CAD	Computer Aided Diagnosis
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
ReLU	Rectifier Linear Unit
MLP	Multilayer Perceptron

1 INTRODUCCIÓN

El presente trabajo describe el proceso de diseño de una aplicación de ayuda al diagnóstico médico. Se fundamenta en el uso de técnicas de *machine learning* y el tratamiento digital de la imagen. El objetivo será ayudar a los profesionales del área de la salud en la tarea de la detección de la displasia anal.

Debido a lo altamente especializada que resulta esta tarea, el diseño de la aplicación pasa por el entendimiento de la enfermedad, del método de análisis clínico y del estudio de los resultados con los que trabajan los profesionales. El análisis de la displasia anal se realiza por inspección visual de la zona. El médico lo realiza por la técnica de la anoscopia o la anoscopia de alta resolución, HRA por sus siglas en inglés, de *High Resolution Anoscopy*. El desarrollo de dicha técnica puede resumirse: el profesional busca ciertos patrones que den información de la situación de la enfermedad y toma una muestra de tejido de la zona de interés para que se analice en laboratorio. Cuando se obtiene el resultado de la biopsia se determina con precisión el grado de la lesión en caso de que exista.

Se desea realizar una herramienta CAD (*Computer Aided Diagnosis*) que pueda determinar dada una fotografía de un estudio de anoscopia la presencia o no de patrones que denoten la existencia de enfermedad.

La forma de análisis conlleva la búsqueda visual de patrones, texturas, colores, protuberancias… Haciendo muy subjetivo el proceso de diagnóstico. Por esto, la asistencia puede ayudar a los médicos a entrenar y ganar seguridad en futuros exámenes a pacientes. El análisis de la displasia anal conlleva una curva de entrenamiento que podría reducirse en dificultad con herramientas como la que el presente trabajo pretende ser.

El uso de técnicas de procesamiento de imagen así como de la inteligencia artificial está entre los métodos de mayor crecimiento en la actualidad, y son objeto de gran interés en aplicación médica. El uso en este trabajo de dichas técnicas se justifica por varias cuestiones. La primera es que el análisis de la displasia anal se fundamenta en la búsqueda visual de ciertos grupos de texturas y colores, de forma que los patrones que resultan relevantes a los médicos son de difícil caracterización. En ese sentido, las técnicas de *machine learning* permiten evitar la búsqueda de las características principales que identifiquen cada patrón ya que se entrena un programa con imágenes de prueba y con un gran número de imágenes de cada patrón, forzando al programa a aprender de forma inspirada en el aprendizaje humano para que en dichas imágenes de prueba, determine la presencia o no de los patrones.

La aparición de lesiones precancerígenas y del cáncer de ano está motivada por la persistencia de genotipos de alto riesgo del virus de papiloma humano en el canal anal. En el caso de personas que sufren el VIH el riesgo de cáncer anal es mucho mayor y especialmente en el caso de hombres que tienen sexo con hombres, HSH. Así, se realiza un seguimiento de personas que se encuentran en las condiciones descritas. El procedimiento clínico consiste en la realización de citología ano-rectal y la toma de biopsias a través de la Anoscopia de Alta Resolución (HRA). La citología muestra un bajo valor predictivo negativo, es decir, se realizan pruebas en las que los resultados afirman la no existencia de enfermedad, pero al realizar pruebas más exigentes y precisas, se descubre que sí existía. Durante la anoscopia, se realiza un examen visual. En el caso de que el médico detecte alteraciones, toma biopsias en dichas zonas. Se trata de un proceso similar a la colposcopía cervical, en la que se trata de analizar el cuello del útero como prueba para detectar el cáncer cervical. En el caso de la anoscopia, se tiene un aprendizaje bastante mas largo y requiere mas de 200 HRA. Así, dicha prueba queda limitada a centros especializados en los que se dispone de un gran número de personas susceptibles de contraer la enfermedad. Esto resulta en una motivación para el desarrollo de este trabajo, ya que un algoritmo capaz de detectar las lesiones en imágenes permitiría a nuevos médicos realizar un aprendizaje mucho más rápido.

El trabajo que aquí se describe está dentro del marco de realización de unas prácticas en la Fundación Pública Andaluza para la Gestión de la Investigación en Salud de Sevilla (FISEVI), donde se desarrollará el proyecto POMPIS, código del Proyecto de Orofotografía Macroscópica Para Imágenes Sistemáticas de displasia en canal anal. Durante el desarrollo de las prácticas, se trabaja en el entorno del Hospital Universitario Virgen del Rocio, colaborando con la Unidad Clínica de Enfermedades Infecciosas, Microbiología y Medicina Preventiva. El último participante es la Universidad de Sevilla, a la que pertenece como estudiante de grado quien escribe estas líneas, concretamente, actúa el Departamento de Teoría de la Señal y Comunicaciones de la Escuela Superior de Ingeniería.

Al encontrarse este trabajo en el entorno real de la investigación y a hombros de un proyecto de mayor alcance y envergadura, debe comprenderse que ciertos objetivos del proyecto POMPIS resultan de mayor interés para este trabajo fin de grado y por tanto, estará centrado en algunos aspectos y tratará menos otros. Aún así, se tratará de introducir completamente el proyecto POMPIS y al menos, describir analíticamente las necesidades y posibilidades de aquellos objetivos que no sean tratados exhaustivamente.

2 DESCRIPCIÓN DE LA ENFERMEDAD

La incidencia del cáncer anal se incrementa desde los años 70[10]. Está fuertemente influenciada por la presencia en los pacientes del Virus del Papiloma Humano, HPV por sus siglas en inglés, *Human Papilloma Virus*[10]. El HPV es el agente mas común de los que resultan ser transmitidos sexualmente[1,2]. El virus del papiloma afecta a las células epiteliales, que son las que recubren las superficies del cuerpo, incluidas las de los órganos y las mucosas. Solamente una pequeña parte de quienes se infectan de HPV desarrolla cáncer, se cree que el HPV es necesario pero no suficiente para el desarrollo de la enfermedad[2]. Así que existe una población de riesgo, que se define por una serie de parámetros que juegan un papel fundamental a la hora de determinar si se acaba desarrollando el cáncer o no. A modo de sistema preventivo, los médicos habitualmente tratan de reducir la incidencia mediante la identificación de las lesiones precursoras al cáncer anal, las HSIL, del inglés *High-grade Squamous Intraepithelial Lesion*, o Lesión Intraepitelial Escamosa de Alto grado, para tratar de remover dichas lesiones antes de que se desarrolle el cáncer[1].

El virus del papiloma humano tiene forma circular, concretamente de icosaedro. Se trata de una familia de virus sin cobertura y de unos 55 nm de diámetro [3]. El principal método de contagio es el contacto directo entre mucosas. Se trata de un virus de tipo dsDNA, es decir un virus cuyo material genético se compone por una doble cadena, de ahí su nombre en inglés, dual string DNA. Los dsDNA son los virus más comunes, son dependientes del ADN para replicarse, en lugar del ARN, usado por otros tipos de virus. Así, este tipo de virus debe introducirse en el núcleo de la célula para replicarse y puede forzar a la célula a mantener constantemente la división celular, produciendo en último momento el cáncer. Todos los papilomavirus paretecen a la familia Papillomaviridae, que tiene 16 géneros que albergan muchos tipos [4]. La estructura genómica del virus se divide en tres regiones: la región de control largo (LCR), que regula la expresión de los genes y la replicación; la región temprana (E), que codifica proteínas necesarias para la replicación y supervivencia del virus, que son genes de expresión rápida, y la región tardía (L) que codifica las proteínas virales estructurales, con genes de expresión lejana en el tiempo[4]. Las regiones E y L están referidas a las fases del ciclo de vida del virus. Las células huésped

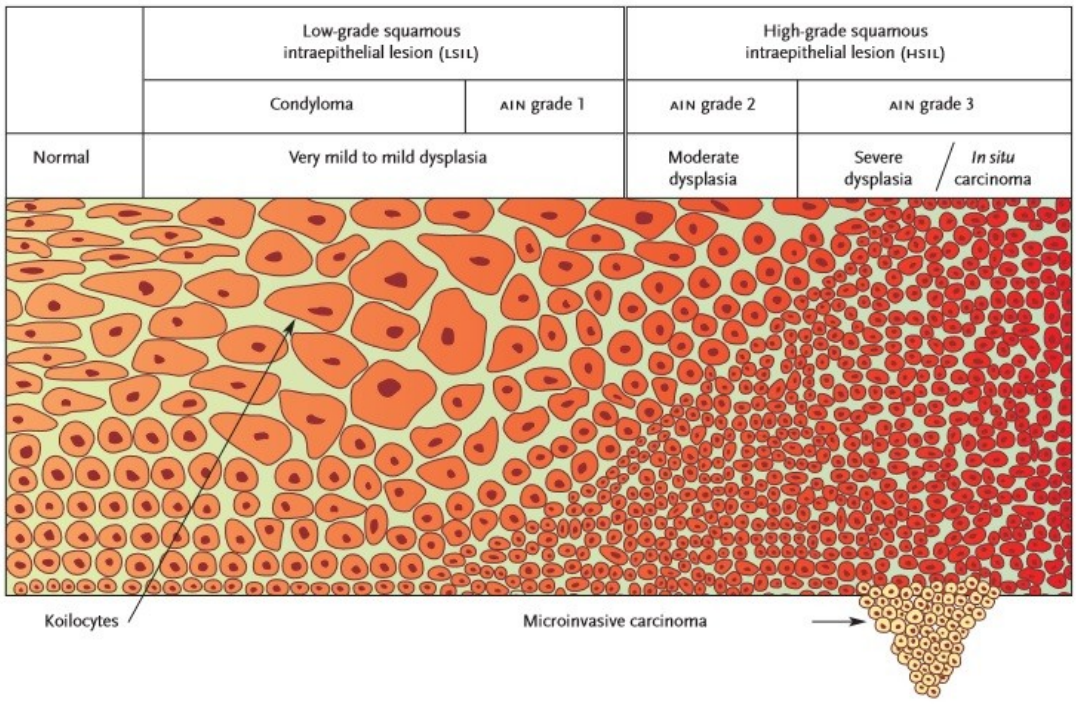


Figura 1: Representación en esquema de las lesiones escamosas intraepiteliales. Extraída de [4]

de los tipos de HPV en humanos son epiteliales, y el virus actúa sobre regiones cutáneas y con mucosas[6]. Como se ha mencionado, el HPV tiene varias grupos, de los cuales destaca el grupo alfa. Contiene éste 64 tipos que principalmente afectan a la mucosa epitelial[3,10]. Unas 40 variantes de esos HPVs pueden infectar el tracto anogenital e incluyen 15 tipos, llamados de alto riesgo, son los HPVs 16, 18, 31, 33, 35, 39, 45, 51, 56, 58, 59, 68, 73, 82 y son causantes del cáncer [3].

El ciclo de vida de los HPVs toma lugar especialmente en zonas de epitelio escamoso estratificado [1,2,3,4],como puede verse en la figura 1. El objetivo del virus es atacar a las células o láminas basales, que están bajo las células epiteliales. Se asume que los virus son capaces de penetrar a través de microtraumas en los tejidos epiteliales hasta llegar a las mencionadas células basales [4]. El desarrollo tanto en el cérvix como en el ano se da en la zona de transición entre mucosas, donde hay mayor vulnerabilidad y donde además, las microlesiones que se producen durante las relaciones sexuales favorecen la infección [1,5,6].

El cáncer de ano tiene establecidos ciertos factores de riesgo [1,3,5]: un alto número de compañeros sexuales, el sexo anal receptivo, un historial previo de enfermedades venéreas, una infección de VIH o el consumo de tabaco. El caso concreto del VIH implica que esas personas tienen mucho mayor riesgo de desarrollar cáncer de ano, de 40 a 130 veces[1]. Siendo de hecho la principal neoplasia en la población infectada de VIH en países desarrollados [1].

Una gran cantidad de esfuerzo en investigación se centra en el HPV16, su ciclo de vida y su rol en el cáncer de cervix y ano. Junto con el HPV16, que es sospechoso de causar el 55% de los cánceres de cérvix [3], está el HPV18 que causa en torno al 15% de casos. Éstos tipos de cáncer son precedidos por lesiones que son resultado directo de la infección por HPV. Con la toma de biopsias o la realización de citologías, se pueden buscar anormalidades en la superficies de los tejidos mucosos[1,3], de forma que se realiza una clasificación y seguimiento de las lesiones previas a aquellas personas que pertenecen a la población de riesgo.

El canal anal consiste en un segmento del tracto intestinal de unos 4 cm de mucosa y células escamosas. Está limitado distalmente por la zona de transición entre la piel y la mucosa y proximalmente por la línea dentada, que es la zona de transición entre la mucosa escamosa y la glandular. Viendo la figura 2, está nuestra región de interés, llamada Zona de Transición (AnZT) situada entre la unión squamocolumnar, SCJ del inglés *Squamo Columnar Joint* y la línea peptínea o dentada LD. Es en esta zona donde los profesionales realizan la observación y seguimiento.

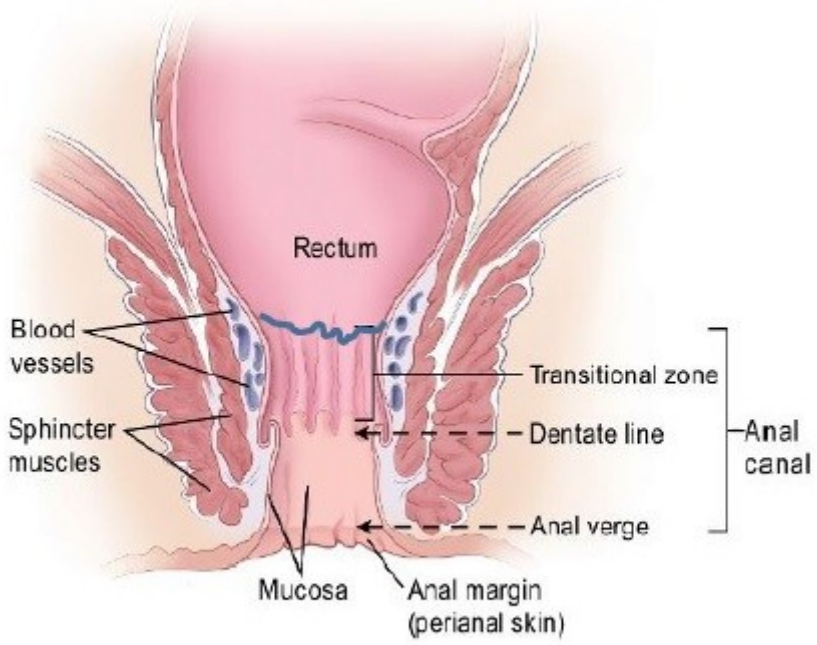


Figura 2: Anatomía del canal anal. Cortesía de Viciana et al [1]

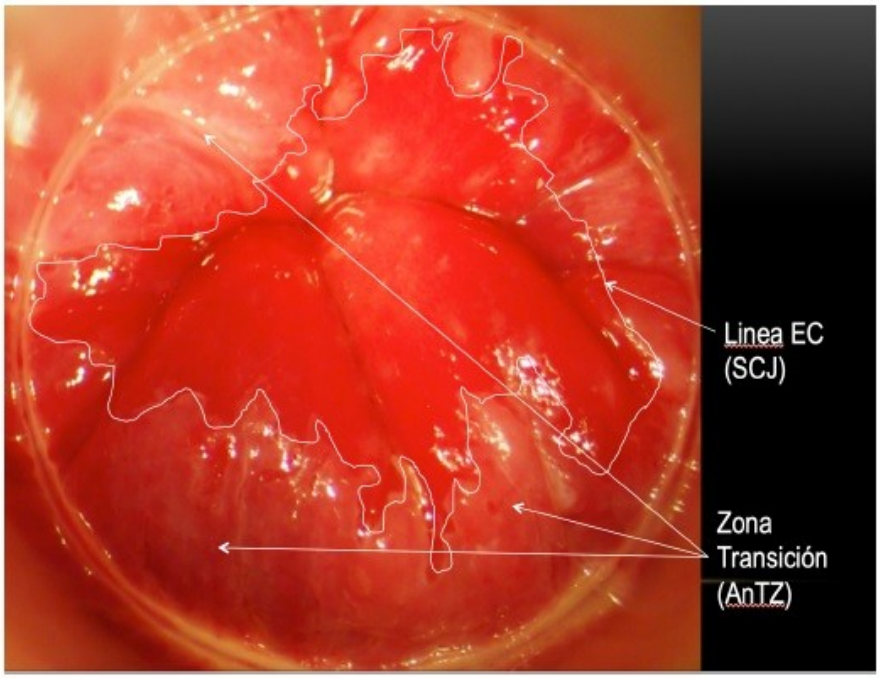


Figura 3: Detalle de la zona de transición. Cortesía de Viciana et al [1]

La figura 3 muestra el aspecto real de la región del canal anal según puede observarse mediante la mencionada técnica de Anoscopia de Alta Resolución. Puede verse la diferencia entre las áreas relevantes. Durante el examen clínico, el profesional busca la zona de transición y la línea escamo-columnar, que presentan una evidente diferencia de mucosas. Y es donde las lesiones tienen lugar.

Según un estudio de 1998 realizado en la Universidad de California, San Francisco, el doctor J. Palefsky buscó infección por HPV en un grupo de HSH y de ellos 346 VIH positivos y 262 VIH negativos. El primer grupo presentó ADN del virus HPV en un 93% de los individuos y el segundo en un 61% [5]. El HPV16 resultó ser el tipo más común en ambos grupos. La presencia de varios tipos se encontró en el 73% de los VIH positivos frente al 23% de los VIH negativos. Con otra cohorte de pacientes seguidos por 4 años, se encontró que un 49% de pacientes HIV positivos desarrollaron HSIL, frente a un 17% de los HIV negativos[5].

Según una revisión de estudios centrados en el tema, los HSH que resultaron VIH+, fueron en mayor medida afectados por el HPV que los VIH-, además desarrollan con más frecuencia anormalidades derivadas del HPV y finalmente cáncer anal. Los resultados del análisis de 53 estudios sugieren un ratio de displasias de alto grado en lo varones VIH+ de en torno al 10%, frente a un 3% de los VIH-[9]. En una vista general de los casos de cáncer de ano, se concluye que el 84% tiene ADN del HPV. El 87% de los individuos infectados de HPV dio positivo para HPV16, y un 6% para el HPV18[9].

Entonces, es habitual realizar seguimiento a los HSH independientemente de su estado respecto a una infección de VIH [5]. Se incluyen otros grupos a considerar: mujeres con cáncer cervical, mujeres y hombres VIH positivos, individuos con condiloma perianal y otros individuos con sistemas inmunológicos comprometidos [5].

La técnica estándar de análisis a los pacientes consiste en realizar una citología, un examen digital y una anoscopia. Durante la citología, se inserta un bastoncillo en el canal anal de forma que quede impregnado en la mucosa contra la que se frota. Una vez insertado, debe moverse y rotarse para capturar las células que serán depositadas en un medio para su análisis en laboratorio. El análisis citológico es un método simple por eso la confirmación de que existe zona displásica requiere de la toma de biopsias a través de anoscopia. La citología anal además, muestra bajos valores predictivos negativos [1]. Los casos de displasia de grado alto pasan a menudo desapercibidos, probablemente porque en la zona del canal anal es mucho mas difícil técnicamente tomar una muestra ciega con el bastoncillo que por ejemplo en el caso del cérvix [9]. En todo caso, si hay alteración, se remite al paciente a HRA para la toma de biopsias de las áreas relevantes del canal anal [1]. La anoscopia se parece a la

colposcopia cervical, pero conlleva un mayor tiempo de aprendizaje.

Figura 4: Aspecto de un anoscopio del catálogo BLife. Fuente: https://www.blife.it/anoscopio-monouso.html

Para realizar la prueba se utiliza el anoscopio, que es el aparato que aparece en la figura 4. Como puede verse, tiene una estructura de tubo que permite ver el interior. Se utiliza un mandil para que el final del tubo, que queda introducido en el canal anal no tenga una superficie cortante y pueda dañar la zona, con lo cual, para introducir el dispositivo, el médico debe meter el mandil en el anoscopio y retirarlo para observar. En el mango se incluye un soporte para una luz que permite ver con claridad la zona. Se utiliza un lubricante para introducir el anoscopio y puede usarse también anestesia para evitar dañar al paciente. El dispositivo de la figura 4 tiene una longitud de 15 cm y un diámetro de 2 cm.

Es posible entonces observar la zona interior del canal anal, que se mostrará rosáceo y rojo además de brillante. Podrán observarse los tejidos y estructuras del interior, normalmente el médico buscará introduciendo o sacando el anoscopio la línea de transición. En esta fase, se introduce un bastoncillo empapado en ácido acético disuelto en agua en torno al 3-5%. Tras aplicar el contraste, las zonas con tejidos displásicos o inflamados se tornarán blanquecinas. Las áreas blancas son después inspeccionadas para buscar patrones que sean sospechosos. El otro tinte que se utiliza es una solución de Lugol, con base de yodo. Con el Lugol, los tejidos sanos se tiñen de negro, caoba, o de un color marrón oscuro. Las lesiones displásicas no absorben bien la mencionada solución y se mostrarán de color amarillo[5]. Con el uso de ambos tintes, los profesionales pueden localizar zonas para concentrar la atención y decidir si biopsiar. Ver figura 5

El tratamiento de las lesiones previas al cáncer tiene varias posibilidades. Existe la posibilidad de retirar los tejidos con cirugía pero lo más habitual es la cauterización. La idea es que si las lesiones de tipo HSIL son precursoras del cáncer anal, realizando una ablación en las zonas en las que las lesiones se encuentran, se prevendrá la progresión del cáncer. Para esto, como se ha dicho, se emplea una termocoagulación.

Una HRA satisfactoria permite ver el ano en su totalidad además de[7]:

- El canal anal completo, incluida la línea escamo-columnar
- La zona de transición AnTZ
- La zona distal del canal anal
- El margen anal
- El periano

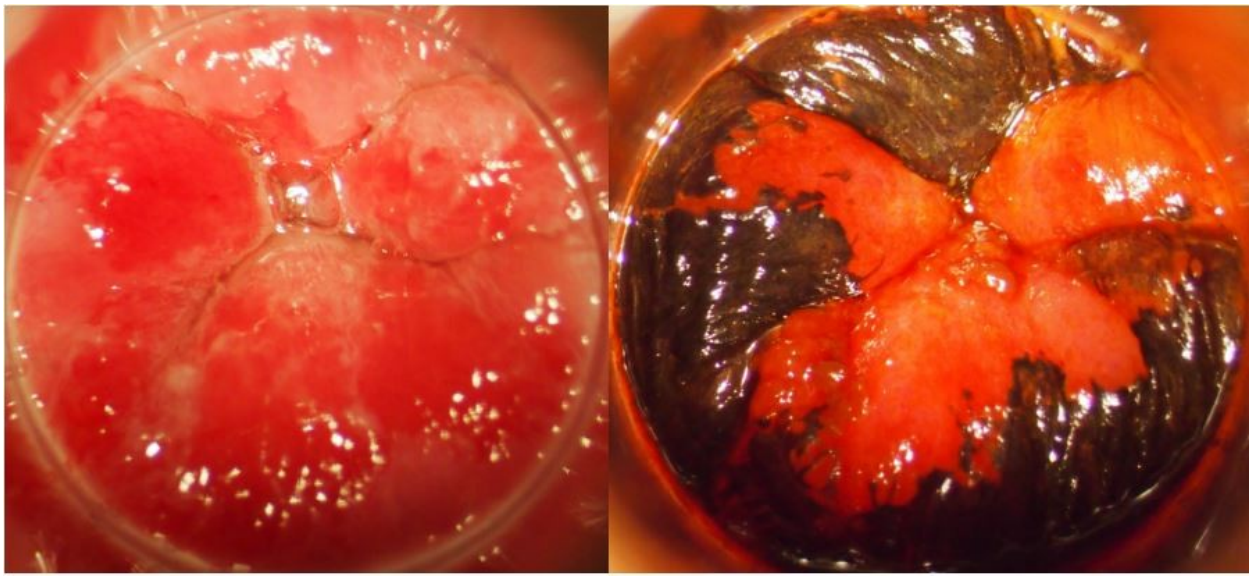


Figura 5: Aspecto de la zona de transición con: ácido acético a la izquierda y con Lugol a la derecha

Se depende en la HRA de la habilidad del médico para reconocer las zonas anormales con el ácido acético y el Lugol, patrones vasculares y características morfológicas y de color[7]. La descripción de las características de la lesión se hace siguiendo una serie de patrones. Una propuesta por la Sociedad americana por la colposcopía y la patología cervical es la siguiente:

- Contorno
  - Plano: sin grosor o elevación sobre los tejidos adyacentes
  - Elevado: con engrosamiento
- Patrones superficiales
  - Suave: superficie igualada y sin texturas
  - Granular: irregular, con la superficie repleta de gránulos
  - Papilas: finas proyecciones
  - Micropapilas: de menor tamaño y mas aplanadas que las papilas
- Patrones vasculares
  - Punteado: se pueden ver los capilares siguiendo un patrón de puntos en la zona
  - Mosaicismo: los capilares hacen bloques irregulares
  - Vasos verrugosos: los vasos sanguíneos presentan anormalidades y protuberancias
- Márgenes
  - Definido
  - Regular
  - Mal definido
- Respuesta al Lugol
  - Negativa: sin señalar la yodina, amarillo
  - Parcial: color variable entre amarillo y negro/marrón
  - Positiva o completa: color negro/marrón uniforme

Los profesionales entonces, para poder localizar y contrastar con otros médicos los resultados, utilizan sistemas de posicionamiento[7]. Debido a que el anoscopio tiene un visor circular, es habitual realizar una división radial del mismo. Suele utilizarse como referencia la postura en la que se realiza al paciente, que normalmente es decúbito lateral izquierdo. Así, en distintos países se recurre a sistemas de localización variados. El sistema dependerá de la posición del paciente, que debe indicarse.

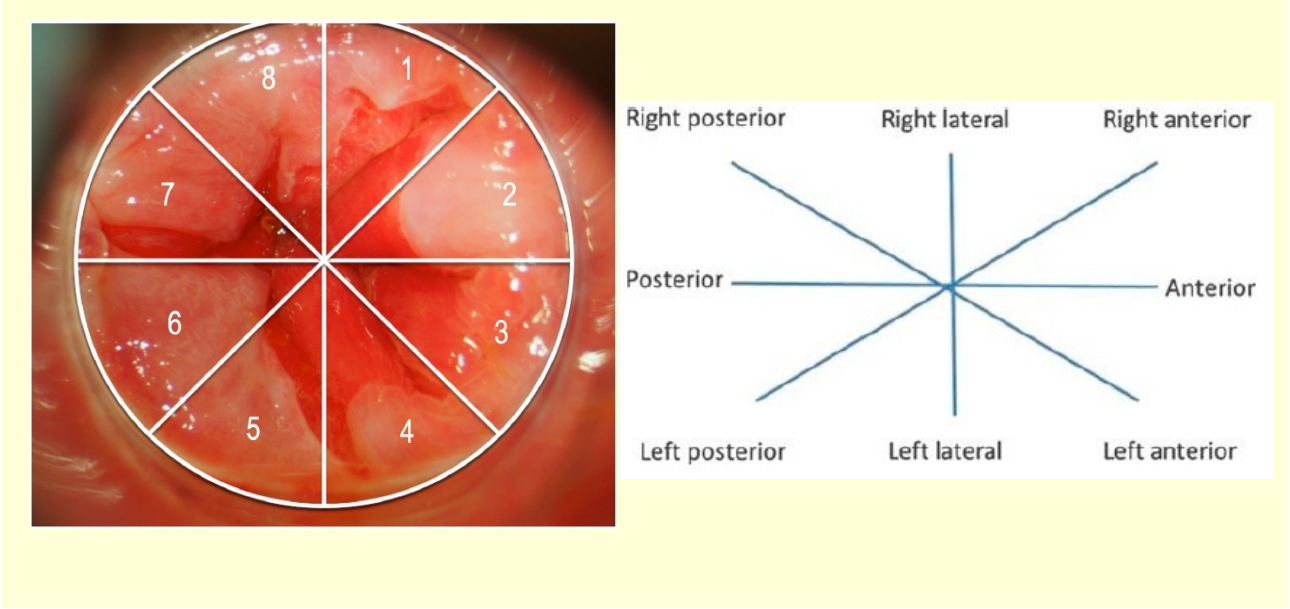


Tabla 1: Formas de localización de las lesiones. Izquierda: uso de octantes numerados, fuente: Viciana et al. [1]. Derecha: uso de octantes con descripción anatómica, fuente Hillman et al. [7]

Como puede observarse en las figuras de la Tabla 1, existen métodos compatibles aunque se echa en falta un criterio común para todos los profesionales internacionalmente, de forma que se facilite el entendimiento de datos de fuentes diversas. También es habitual el uso de la división en 12 secciones con inspiración en los sistemas de orientación de los relojes. Se sitúan las 12 justo entre los octantes 6 y 7, apuntando a la zona posterior del paciente. Tendría que imaginarse un reloj sobre el extremo del anoscopio con las 12 apuntando a la espalda. Como se ha comentado, los pacientes se deben colocar en la misma postura para poder tomar las fotografías desde una posición mas o menos fija. Así, al estar tumbado sobre su lateral izquierdo: los octantes 6 y 7 y la zona posterior apuntarán a su espalda, y serían las 12 del reloj; los 2 y 3 y la zona anterior a los genitales y serían las 6 del reloj.

Así, es posible contrastar el progreso de la enfermedad al tener dichos descriptores de posición. En cada nueva inspección se pueden tener señaladas las áreas concretas que fueron revisadas y analizadas previamente y localizar la toma de biopsias para contrastar el diagnóstico inicial que motivó la toma con el resultado final.

En el estudio realizado por Camus *et al.* 2015, se intenta hacer una comparación entre los patrones presentes en las lesiones precursoras del cáncer anal para ver cuáles resultan más habituales en las displasias de alto y bajo grado. Se tienen 168 lesiones biopsiadas de 103 pacientes y se comienza afirmando que un 58’6% de lesiones de bajo grado y un 57’1% de lesiones de alto grado resultaron no ser visibles al ojo[8]. Un 41’7% de las lesiones son de alto grado, un 34’5% de bajo grado y un 23’8% son no displásicas. Las lesiones con patrones epiteliales y vasculares irregulares resultan tener el doble de probabilidad de acabar siendo de alto grado comparadas con las lesiones con patrones epiteliales y vasculares regulares[8]. La incidencia del blanqueamiento por ácido acético se sitúa en un 91’4%, 94’8% y 70% para las lesiones de alto grado, bajo grado y no displásicas. Y de entre las lesiones de alto grado, un 62’9% presentan respuesta negativa con el Lugol frente a un 31% de las de bajo grado.

Se tiene también que comparando las lesiones que resultan en displasia y las que no: los patrones irregulares epiteliales, los cambios vasculares, el blanqueamiento con ácido acético, los patrones punteados y en mosaicos están significativamente más presentes en las primeras que en las segundas[8].

Comparando las lesiones de alto grado y las de bajo grado, las lesiones de alto grado presentaron lesiones irregulares epiteliales y vasculares en un 68’6% y un 61’4% frente al 37’9 % en ambos casos para las lesiones de

bajo grado[8].

Además, únicamente un 38’7% de las lesiones fueron detectables a la vista, destacando así la necesidad de utilizar la HRA para el diagnóstico de las mismas [8]. Se intenta concluir que el ácido acético no presenta una relación de exclusividad con las lesiones de alto y bajo grado, ya que las no displásicas también se blanquean, como los condilomas, las hemorroides o las lesiones inflamadas. Se destaca también la necesidad de seguir generando estudios acerca de los patrones o grupos de patrones que mejor caracterizan los diferentes grados de lesión.

La prevalencia de la displasia anal en HSH parece no decrecer con la edad, en contraste con las lesiones de cérvix [9] . Esto puede estar motivado en parte por los comportamientos y hábitos sexuales de dichos grupos. En un estudio realizado en Australia, un 38% de HSH reportó haber tenido más de 50 parejas con las que mantuvieron relaciones sexuales, de las mujeres heterosexuales estudiadas, únicamente un 1% llegó hasta esa cifra. Esto podría justificar las diferencias que con la edad se encuentran entre la persistencia del HPV y sus problemas derivados en HSH y en mujeres[9].

Se debe decir, que aunque es habitual comparar el desarrollo del HPV en el ano y en el cérvix, no pueden extrapolarse los criterios de análisis clínicos [1,3,9]. Algunos autores tratan de justificar procedimientos por su efectividad en el estudio del cérvix que en el ano resultan algo más complejos, por ejemplo las mencionadas dificultades con las citologías.

Entonces, el desarrollo de un programa de estudio y seguimiento del cáncer anal es complejo. La identificación de patrones o marcadores biológicos para establecer que pacientes con altos grados de displasia tienen mayor riesgo de desarrollar cáncer de ano debería ser una cuestión prioritaria [9]. Así podría distinguirse entre los individuos que probablemente se recuperen de la lesión y cuales necesitan mayor atención por guardar altas probabilidades de que la enfermedad se siga desarrollando.

3 ESTADO DEL ARTE

El ámbito de la sanidad y la medicina se generan enormes cantidades de datos [11]. Habitualmente se trata con datos recogidos con equipos de gran precisión y calidad. La generación de algoritmos para ayudar a superar las dificultades con las que los profesionales lidian se nutre de tales cantidades de datos, hoy en día es especialmente notorio con herramientas como la inteligencia artificial o AI.

Casi cualquier tipo de profesional del ámbito sanitario usará AI en el futuro y en particular el Deep Learning[11]. Mientras que las bases de la AI se remontan muchos años atrás, desde los primeros conceptos puestos en valor por A. Turing, W. McCulloch y W. Pitts, no es hasta la actualidad que el *Deep Learning* se ha popularizado y aceptado ampliamente[11]. Una red neuronal de deep learning es un modelo que permite detectar características de una serie de datos de entrada para acabar generando una serie de datos de salida. Las redes neuronales pueden ser diseñadas para realizar diversas tareas específicas, pueden trabajar con datos como: texto, audio, imágenes, video, datos numéricos… Los resultados de la red neuronal serán predicciones que deben compararse con los objetivos deseado, en el ambiente sanitario serán los diagnósticos de los médicos, para determinar la eficacia del funcionamiento[11].

Para la detección de lesiones precancerígenas en el caso del cáncer de ano apenas existe documentación que trate de resolver la identificación y el diagnóstico con sistemas de aprendizaje profundo. Algunas alternativas similares son fuentes de información indirecta como aquellos trabajos que estudian las lesiones previas al cáncer de cérvix[1,,2,11,12]. Tradicionalmente éste ha sido influyente en la forma de tratar el cáncer de ano, pero cada vez más, los profesionales intentan separarse de esas técnicas ya que son solo algunas las características que ambas enfermedades comparten. A pesar de esto, como hay mas trabajos y proyectos, son para este trabajo fuente de información aquellos sistemas de detección automatizados centrados en las lesiones previas del cáncer de cérvix.

El caso del estudio de las lesiones previas al cáncer de ano se basa en el uso de imágenes[1,2,11,12]. Debido a que se trabaja con fotografías que incluyen lesiones en la epidermis, otro caso relevante es el estudio con técnicas de *deep learning* del melanoma. Es una buena fuente de información porque su estudio está mucho mas popularizado que caso del cérvix o el ano. Una muestra de esas imágenes se puede ver en la tabla 2.

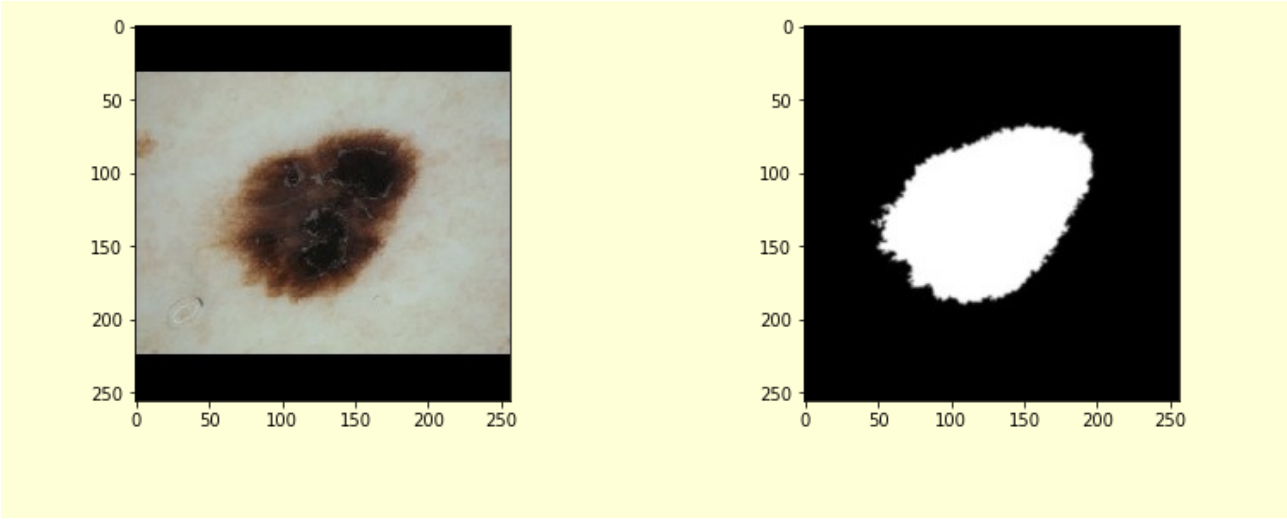


Tabla 2: Muestra de una lesión de piel a la izquierda y máscara segmentada a la derecha.

El gran reto que asume la inteligencia artificial es resolver problemas que los humanos pueden completar de forma sencilla pero que resultan muy difíciles de plantear y describir formalmente[14], como reconocer una voz o

un rostro. En el *Deep Learning* se trata de conseguir que los computadores traten los datos definiendo una relación jerárquica de conceptos. Cada nuevo aspecto de los datos que el modelo pueda encontrar, deberá ser relacionado con otros de mayor complejidad o abstracción. Así, la jerarquía permite al computador aprender conceptos complicados desde otros más sencillos[14]. Si se dibujara un gráfico mostrando esos conceptos apostados unos sobre otros, tendríamos un grafo de muchas capas de profundidad, he aquí la idea del término *Deep Learning*[14,15].

El estudio de imágenes médicas con técnicas de *Deep Learning* e inteligencias artificiales es actualmente de gran interés. Por ejemplo, existen estudios que indican que las lesiones y anormalidades en casos de cáncer de cérvix pueden ser diagnosticadas con mayor acierto usando técnicas de análisis de imagen por computador[12]. Con el uso posterior de redes neuronales, máquinas de vectores de soporte (SVM) u otras herramientas, se pueden clasificar datos de las lesiones previas al cáncer[12]. El uso de las redes neuronales es frecuente debido a sus mejores resultados y potencial para aplicarse en tiempo real[12].

Para llegar al *Deep Learning*, hay que pasar por otros conjuntos de modelos y herramientas. Primero, la Inteligencia Artificial, que contiene al *machine learning*. Dentro del último, está el *representation learning* desde el cual podemos acceder al *Deep Learning*.

El *representation learning* es un compendio de métodos que permiten a una máquina trabajar con datos no procesados encontrando la representación necesaria para detectar características o clasificarlos[12,13,14]. Los métodos de *Deep Learning* pertenecen a los *representation learning*, con la diferencia de que existen múltiples niveles de representación. En el *Deep Learning* se va haciendo cada vez una representación mas abstracta de los datos, agrupando elementos muy simples para poder seguidamente acceder a bloques de información de gran relevancia[13,14,15]. El ejemplo clásico del *representation learning* es el autoenconder. En el autoenconder hay dos funciones, un codificador que transforma la información y la representa de manera distinta y un decodificador que trata de tomar una representación y convertir los datos a la representación original[14].

De entre los métodos de trabajo que el *machine learning* plantea, el más habitual es el *Supervised learning*[13]. Se computa una función que mide el error entre la salida y el valor deseado para cada entrada al sistema, haciendo que la máquina ajuste sus parámetros internos para reducir dicho error[13]. Esos parámetros se llaman pesos, y se pueden almacenar como valores numéricos de una matriz. Desde los años 1990 las técnicas de *Supervised learning* se han ido haciendo cada vez más populares[15] y han sido causa de la enorme proyección que el *Deep Learning* hoy en día tiene.

Las herramientas de *Deep Learning* superan una gran limitación a la hora de trabajar con datos no procesados[12,13]. Crear un clasificador requería la presencia de muchos profesionales de cada sector para poder caracterizar correcta y eficientemente cada clase. Además requería una gran cantidad de horas de diseño para producir un algoritmo sólido que reconozca todos los aspectos relevantes que hacen a un objeto pertenecer a una clase determinada. El ejemplo clásico del *Deep Learning* sería el modelo del perceptrón multicapa (MLP de *Multilayer Perceptron*). Puede imaginarse como una función matemática que mapea valores de entrada a valores de salida. Se compone de muchas funciones sencillas y busca la mejor representación, deduciendo cuál con la estructura de abstracción creciente. Las computadoras deberán aprender cuáles son las características intrínsecas de los datos que mejor los representan en función del problema que se intenta resolver[16]. El modelo de mayor éxito para el análisis de imágenes dentro del *deep learning* es el de las Redes Neuronales Convolucionales (CNN, de *Convolutional Neural Network*)[15]. En dichos modelos, se tienen muchas capas con filtros de convolución que pueden aprender gran cantidad de características que permiten describir un patrón en una imagen. Antes del enorme desarrollo que estas técnicas han tenido, se trabajó con sistemas que buscaban características especificadas por los diseñadores, haciendo que problemas complejos fueran enormemente costosos de modelar[16].

En un trabajo realizado para la detección de las lesiones precancerígenas en el cérvix, se propone el uso de una red neuronal artificial [2]. En ese caso, disponen de un conjunto de datos de imágenes de 170 con las que tratan de encontrar dos clases, una con patrón punteado y otra sin patrón. Son los capilares los que forman los patrones. Aquellos casos bien delimitados y con formas suaves indican gran probabilidad de presencia de una lesión de bajo grado, especialmente si el espacio intercapilar es pequeño. Los patrones irregulares y con formas abruptas

representan una mayor probabilidad de ser lesiones de alto grado[2].

Su propuesta es usar una red neuronal híbrida, que usa un MLP (*multilayer perceptron*) junto con un mapa auto-organizado[2]. El primero es un ejemplo de red que se debe entrenar de forma supervisada y el segundo, del tipo no supervisado. Con el uso combinado de ambas, se trata primero de concentrar la información y detectar qué información es la que mejor categoriza una entrada como perteneciente a una clase concreta.

En la arquitectura de dicho trabajo, la red de Kohonen o auto-organizada tiene tres capas de 64 neuronas cada una, y la salida del mapa auto-organizado sirve de entrada al MLP. La red del perceptrón multicapa tiene 3 capas con 64 neuronas, 45 neuronas y 1 neurona en cada una [2]. Debido a que proponen un clasificador, la última capa debe representar la predicción que la red hace sobre los datos de entrada. De esta forma, esa última neurona tiene dos estados, activada y no activada de forma que en así puede designar la entrada como perteneciente a una clase u otra. Este caso de clasificador binario deberá tener una salida de una neurona, sin embargo un clasificador que trabajara con 1000 clases necesitará en su última capa 500 neuronas.

En el trabajo mencionado, se consigue alcanzar una precisión con la red neuronal de un 72.5%[2], incluso con un modelo de red tan sencillo como el comentado antes,y con muy pocos datos para entrenar. En otros casos, la detección de lesiones por el uso de imágenes y la inteligencia artificial puede desarrollar una precisión de un 95% como en el caso de trabajos realizados para el diagnóstico del cáncer de mama [2]. Esto deja patente la prometedora propuesta que es el uso de las redes neuronales.

Otro trabajo propuesto por Claude *et al.*[17] utiliza una red MLP para clasificar imágenes colposcópicas en base a los contornos de las estructuras. Determinan dos clases de contornos, emborronados o mal delimitados, que tienen pequeñas regiones en los bordes y contornos finos, bien delimitados[17]. Estas imágenes son objeto de tinción con una solución de Lugol. Disponen de 283 muestras de bordes y contornos. Seguidamente realizan entrenamientos con la red MLP que usa funciones de activación sigmoides y rectificadores lineales[17].

En un proyecto realizado en el Albert Einstein College of Medicine de los Estados Unidos se trabaja con redes neuronales artificiales para reducir la tasa de falsos positivos en los frotis realizados durante el examen médico[18]. Se trabaja en dicho proyecto con 487 imágenes catalogadas y clasificadas con los resultados de las biopsias, que fueron de lesiones displásicas precancerígenas de alto grado[18].

No se encuentran otros documentos o publicaciones científicas que traten el tema del tratamiento de imágenes de anoscopia de alta resolución con técnicas de deep learning que puedan ayudar a nutrir la documentación de este capítulo. Como se ha descrito la enfermedad y cómo se trata actualmente, se presenta en el siguiente capítulo una introducción a las técnicas

4 INTRODUCCIÓN A LA TECNOLOGÍA DE DEEP LEARNING

Esta introducción tiene como fuente de información los siguientes trabajos: el libro de Goodfellow *et al* de 2016[14], la revisión de LeCun *et al*.[13], la revisión de Shrestha *et al*. [19] que también ha influido en la clasificación y en la estructura del capítulo, la revisión de Schmidhuber [20] y en el recurso web de deeplearning.net.

En el *deep learning*, la unidad básica de control de la información es la neurona. Existe entre esta tecnología y la neurología una serie de similitudes que se explotan en ambos sentidos. Las estructuras de razonamiento del cerebro de los seres vivos inspira la arquitectura y funcionamiento de los modelos y los algoritmos de aprendizaje. Con el estudio de estas herramientas, que por inspiradas que estén, se distancian mucho de la enorme complejidad del cerebro, son útiles para entender cómo se aprende. Al tener una máquina sencilla con movimientos limitados y sujeto a unas ciertas reglas inspiradas por la propia naturaleza de la tarea, se consigue a veces entender qué hay en esos trabajos que el ser humano pasa por alto pero que es crucial para nuestro propio entendimiento del mundo.

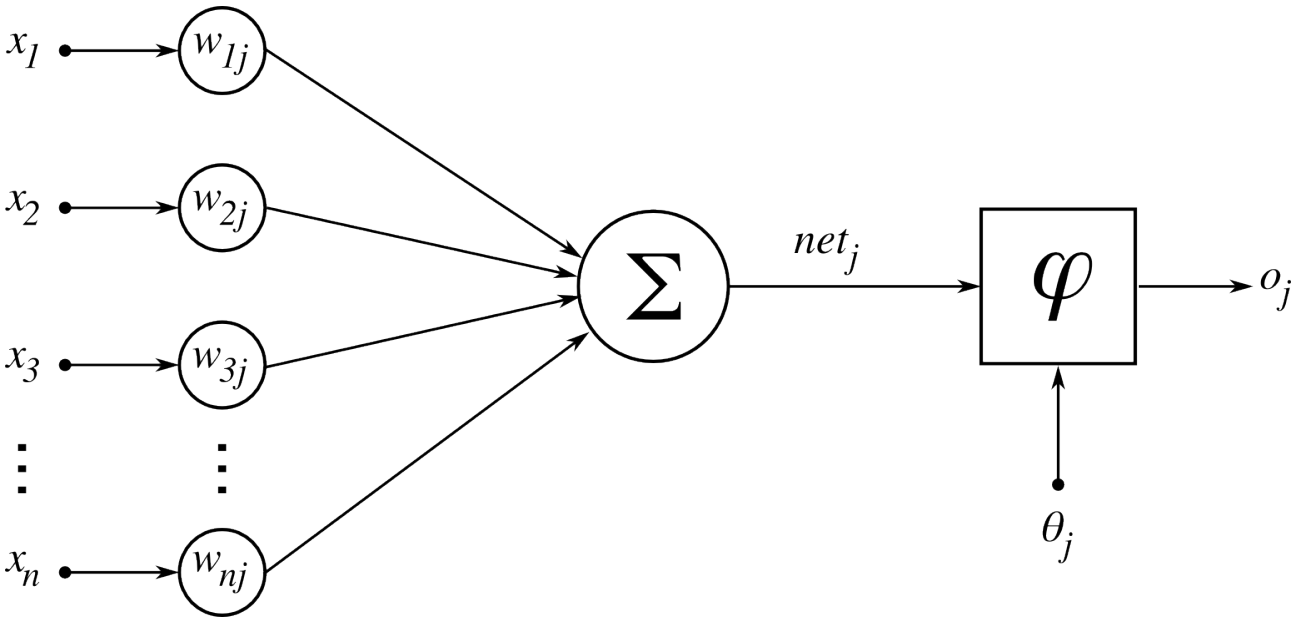


Figura 7: Representación del modelo neuronal utilizado en deep learning. Fuente: https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel.png

Como puede verse en la figura 7, la mencionada neurona o nodo se compone de varios elementos. Puede tenerse en mente también la figura 6, en la que quedan representadas varias neuronas formando capas. Las neuronas entonces se pueden organizar en capas en las que pueden existir todos los nodos que se deseen al diseñar un modelo.

Como se ve en la figura, se toman las entradas $(x_1,x_2,...,x_n)$ y se procesan multiplicándose por una serie de valores agrupados en un vector y que se llaman pesos, W_j por ser los pesos de la capa j. Todos esos valores se suman en la neurona. Este proceso recuerda enormemente al funcionamiento de las unidades básicas del cerebro: las neuronas abren o cierran sus canales iónicos en función del estado de las neuronas colindantes, que influyen en el proceso de activación de cada nodo. Cuando una neurona se encuentra con una diferencia de voltaje suficiente en relación con el exterior de su membrana celular genera un potencial de acción que se propaga a lo largo de la red nerviosa que canaliza dicho impulso y reacciona en función del estímulo recibido. En esta tecnología, se tienen herramientas que emulan ese comportamiento. Cuando todos los pesos se suman en el nodo, se realiza la siguiente operación:

$$y(x) = \varphi(\sum_{i=0}^{n} w_{ij} x_{ij})$$

De esta forma, una neurona debe sumar todos los valores de las entradas por los pesos de las neuronas conectadas a ella misma y luego procesar este resultado con una función de activación, denotada por la letra griega phi. Aquí se utiliza otro valor extra: θ_j que no es mas que un umbral de activación. Este último paso es la forma de controlar o modelar, por seguir con la analogía del cerebro, los potenciales de acción. Estas funciones son normalmente suaves, derivables y la capacidad de comprimir los valores de entrada en un rango que puede controlarse eligiendo otras funciones de activación. La función de activación mas utilizada es la ReLU o unidad rectificadora lineal, como alternativas se tienen también la tangente hiperbólica y la sigmoide.

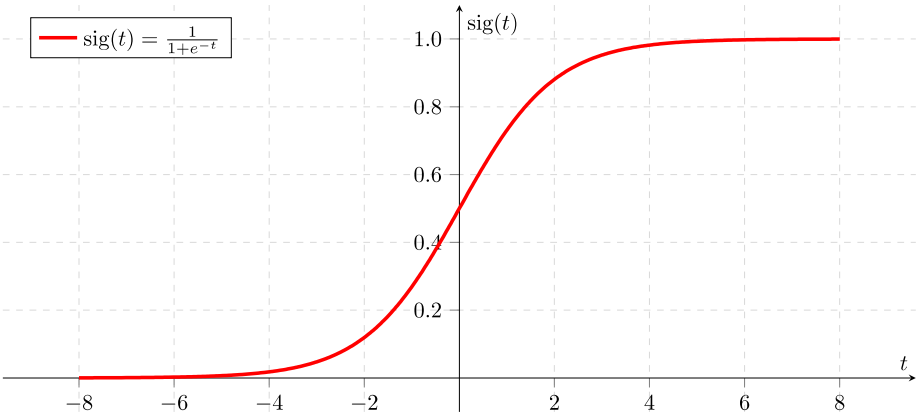


Figura 8: Función de activación sigmoide. Fuente: https://upload.wikimedia.org/wikipedia/commons/5/53/Sigmoid-function-2.svg

Las neuronas tienen algunos elementos mas que se han retirado del modelo anterior por simplicidad, pero uno en concreto debe comentarse antes de ver el funcionamiento de las neuronas en el entorno de una red. Este elemento se llama *bias* en inglés. Es útil para descentrar las funciones de activación. Normalmente la suma de los pesos por las entradas de las neuronas de las capas previas no tiene que quedar situada en la zona central o de interés de la función de activación. En el caso de la figura 8 por ejemplo, se tiene como región de cambio entre el valor -5 y el 5 y retornará una activación segura desde casi el valor 4, pero ¿y si fuera deseable que una neurona concreta se active con un valor 0?, para ello, o se cambian las entradas de la función de activación o se desplaza esta misma. Como las entradas tienen valores que son desconocidos y los pesos van actualizándose durante el entrenamiento de la red, los valores de bias se varían para conseguir que la activación se sitúe justo entre los límites de interés de la suma de pesos y entradas que a cada neurona lleguen.

Las redes *deep feedforward networks*, o *multilayer perceptrons* son los modelos que recogen de forma más general la esencia del *deep learning*. El objetivo de estos modelos es aproximarse a una función f^*. En el caso de un clasificador, $y=f^*(x)$ establece una entrada x en una categoría o clase y. Una red *feedforward* trabaja con una función $y=f(x;\theta)$ y aprende los valores de θ que mejor aproximan la función. Los modelos se llaman *feedforward* o "alimentados hacia adelante" ya que la información fluye desde la entrada hacia la salida, sin conexiones en el sentido opuesto, sin *feedback*. Las redes convolucionales de reconocimiento de objetos son un ejemplo de estos modelos.

La razón de que sean llamadas redes, es porque se representan aglutinando muchas funciones diferentes. Por ejemplo se pone la cadena de funciones siguiente: $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$ donde la función $f^{(1)}$ será la primera capa o capa de entrada de la red, la función $f^{(2)}$ será la segunda capa y $f^{(3)}$ la tercera, en este caso la capa de salida. La longitud de la cadena será la profundidad del modelo, en inglés *deep*, término que da nombre a la técnica. Durante el entrenamiento, la red lucha por alterar y usar las capas para que dada una entrada x se consiga predecir un valor lo más cercano a y posible, consiguiendo así aproximarse a f^*. Como las capas entre la primera y la última son interiores, no se muestran las predicciones y se llaman por esto capas ocultas. Por último, estas redes son llamadas neuronales por la inspiración en las ciencias cognitivas y neurológicas. Cada capa oculta se maneja como una matriz o un vector y la dimensión de dicha estructura será la anchura del modelo. Cada elemento de la capa es una unidad análoga a una neurona que actúa en paralelo con el resto de las de la capa. La analogía con la neurona parte de que recibe una entrada proveniente de muchas otras unidades y computa su propia activación. El usar

muchas capas con forma de vector también bebe de la neurociencia. Sin embargo el objetivo no es modelar el cerebro.

Si las transformaciones que las capas ocultas realizan sobre los datos fueran todas lineales, el conjunto de la red produciría como salida una función lineal de la entrada. Para evitar esto, se usan funciones no lineales para describir las características. Es habitual realizar una transformación afín seguida de una función no lineal llamada función de activación. En las redes neuronales actuales es una recomendación usar inicialmente o por defecto una unidad rectificadora lineal o ReLU, de *rectifier linear unit.* Se define como $g(z)=max\{0,z\}$. Es utilizada una función de activación para computar los valores de los pesos de las capas ocultas o interiores de la red.

Cada conexión de la red tiene asociado un valor o peso. Las entradas se multiplican por los pesos de activación en todas las neuronas de una capa y en la siguiente se suman los valores de todos los pesos de las conexiones que apuntan a cada una de las neuronas de dicha capa. Es ese valor el que finalmente es procesado por la función de activación. Las funciones alternativas a la ReLU son la sigmoide o la tangente hiperbólica, que codifican los valores de entrada en salidas contenidas en los rangos [0,1] y [-1,1] respectivamente. El resultado de dicho proceso es entonces procesado por el nodo de la siguiente capa como una entrada y se repite el proceso a lo largo de la red y a medida que varían los valores de los pesos durante el proceso de aprendizaje.

La salida final de la red puede ser muy variada, por ejemplo puede ser un valor numérico que se corresponda con una lista de clases si la tarea consiste en clasificar, o los distintos valores de tres matrices de dimensiones concretas para formar una imagen a color, o una cadena de texto…

Vistazo rápido a la tecnología

Un gran número de algoritmos de *deep learning* se pueden describir de forma simplificada con los siguientes elementos:

- Un conjunto de datos
- Una función de costes
- Un procedimiento de optimización
- Un modelo

El conjunto de **datos** debe ser una representación de la información que se dispone para resolver el problema. Normalmente es deseable poder trabajar con una gran cantidad de datos, ya que así se pueden estudiar muchas situaciones y generalizar una solución.

La **función de costes** es una herramienta que permite al algoritmo mejorar su funcionamiento. Normalmente se intenta reducir las pérdidas que la función de costes dicta y esto provoca el incremento de la precisión.

El **procedimiento de optimización** se refiere a la forma en la que el algoritmo gestiona los cambios en sus parámetros internos para que dadas ciertas entradas se localice la configuración que con mayor eficacia consigue mejorar la función de pérdidas.

El **modelo** es la arquitectura de capas y conexiones así como de operaciones que los datos atraviesan hasta que se genera una predicción. Se refiere a los procesos que toman lugar durante el trabajo del algoritmo. Ciertas arquitecturas son mejores para realizar unas tareas, como por ejemplo el reconocimiento de patrones en imágenes y otras tienen mejor rendimiento en el campo de la predicción de texto.

De esta forma, si se tiene un sistema con los elementos descritos y se realiza un entrenamiento, se tiene que un **modelo** procesa unos **datos**, acorde a un algoritmo de aprendizaje compuesto de entre otros elementos una

función de costes y un **proceso de optimización** de forma que se encuentre la configuración de parámetros del modelo que con mayor acierto sea capaz de **predecir** para nuevos datos, la salida que corresponda.

Cuando un modelo procesa suficientes datos, habrá ajustado sus parámetros internos lo suficiente como para ser capaz de dado un nuevo dato, encontrar la predicción correcta. En esta situación se dice que el modelo ha **generalizado** una solución. Esta generalización se refiere a una buena capacidad de trabajo por parte del algoritmo con datos que no se han procesado previamente. Para esto, se trabaja con conjuntos de datos que están separados, uno para entrenar y otro para las pruebas. Cuando el algoritmo entrena utiliza el primer conjunto, y para comprobar que se ha generado correctamente se debe utilizar el segundo. Esta división debe hacerse asumiendo que ambos conjuntos están distribuidos idénticamente y unas muestras son independientes de las del otro grupo. De esta forma se despliegan dos objetivos enormemente deseables:

- Que el error durante el entrenamiento, con el conjunto de entrenamiento sea pequeño.
- Que la diferencia entre el error en el conjunto de test y el error en el conjunto de entrenamiento sea lo menor posible.

Así se puede intentar analizar el comportamiento del algoritmo. Y estos parámetros anteriores son los que llevan a dos puntos importantes del *machine learning* y el *deep learning*, el **sobreajuste** u ***overfitting*** y el ***underfitting*** o **subajuste**. El primero se da cuando la diferencia entre el error de test y el de entrenamiento es muy grande y el segundo cuando el error de entrenamiento es, de nuevo de considerable valor.

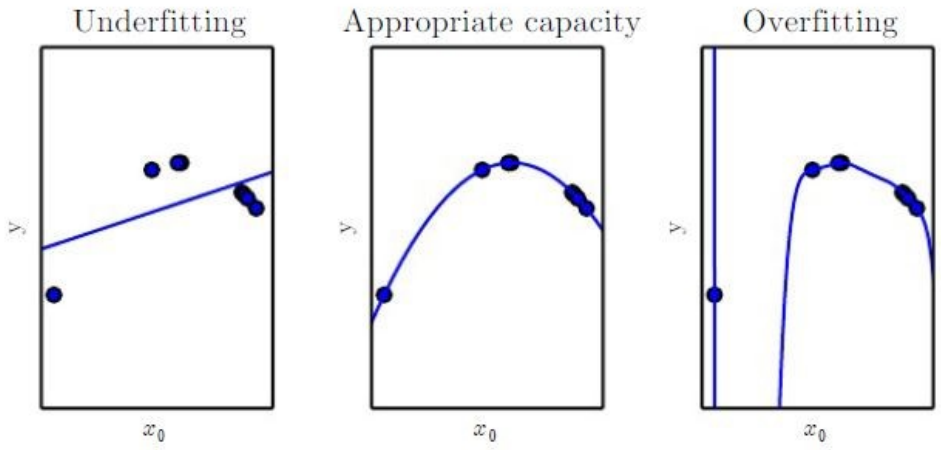


Figura 9: Comparación entre el sobreajuste y el subajuste. Fuente: Goodfellow et al. 2016

En la figura 9 puede verse la importancia de la **capacidad**. La capacidad de un modelo es la clave para poder controlar el sobreajuste y el subajuste. La capacidad se refiere a todas las funciones que el algoritmo puede realizar. Un modelo con poca capacidad puede tener muchos problemas para procesar el conjunto de entrenamiento y un modelo con demasiada capacidad podría memorizar casos en lugar de aprender las características que hacen a los datos tener una forma concreta. En la mencionada figura se compara el funcionamiento de modelos (de izquierda a derecha): linea, cuadrático y de orden 9 para que encuentren la función que mejor describe un conjunto de datos representado por los puntos azules cuya distribución corresponde a una cuadrática. La mejor estimación que cada modelo da es la línea azul de cada solución. Cuando el modelo tiene poca capacidad, en el primer caso, el modelo busca la recta que mejor define una curva, se trata de un caso claro de subajuste. El caso del medio representa la situación ideal en la que se dispone de un problema y un modelo con capacidades similares, que produce una solución que no solo es capaz de resolver el problema sino que lo hace de forma eficiente. El último caso, en la situación de sobreajuste se tiene un sistema que es capaz de dar una solución pero no encuentra la mejor representación de los datos. Como se está haciendo una predicción de un sistema de orden 2 con otro de orden 9 se está trabajando con un sistema que puede dar una solución o familia de soluciones que no recojan el verdadero comportamiento de los datos. Puede verse esto representado por el valle que existe en la función que el último modelo predice como solución.

Como contrapartida del sobreajuste, la convergencia temprana que ocurre con el subajuste lleva a la no generalización de una solución. Estos son los dos principales problemas a resolver durante el trabajo con redes neuronales.

Dada esta primera pincelada, que resulta algo intuitiva, se presenta ahora una clasificación generalizada de los algoritmos de *deep learning*.

4.1. Clasificación de redes neuronales

Las redes neuronales pueden dividirse en dos grupos, los modelos condicionales o *discriminative* en inglés y los modelos generativos. El primer conjunto se utiliza en el aprendizaje **supervisado** y el segundo en el aprendizaje **no supervisado**.

El aprendizaje supervisado requiere de etiquetas para los datos que se usan en el entrenamiento de la red. Esto hace que la red busque la solución que mejor representa los datos en función de los criterios que se generan por la forma en la que las etiquetas son generadas, y por el mismo comportamiento que las etiquetas describen. Por ejemplo, el caso que nos ocupa en este proyecto es de aprendizaje supervisado porque se trabaja con imágenes para las cuales se dispone del resultado de una biopsia, esto da una **verdad de referencia** que la red debe aprender a descubrir en aquellos nuevos datos que le sean enviados para procesar.

El aprendizaje no supervisado no cuenta con estas etiquetas. Explota las similitudes entre los datos para generar información útil.

Se tienen los siguientes tipos [19]:

- *Feedforward Neural Networks* o MLP (*Multilayer Perceptron*). Son las redes de mayor sencillez. Se trata del tipo descrito al inicio de esta sección. La información viaja en un sentido, de la entrada a la salida, sin realizar bucles o vueltas.

- *Recurrent Neural Networks* o RNN. Estos tipos de redes tienen bucles entre sus conexiones internas. Concretamente, la capa de salida se convierte en la entrada de la siguiente capa, si las RNN suelen tener una única capa, la salida se vuelve a procesar consiguiendo un bucle realimentado. Es una arquitectura útil para tomar entradas que son dadas en secuencia y para las que es necesario ir actualizando o variando el comportamiento, e.g. reconocimiento de voz.

- *Radial Basis Function Neural Networks*. Tiene la estructura típica de capa de entrada, capas ocultas y capa de salida. Este tipo de redes trata de aglutinar la información o las características en torno a una función de base radial para la que en cada nodo de cada capa se tiene un posible centro.

- *Kohonen Self Organizing Neural Networks*. Este tipo de redes auto-organiza la arquitectura de la red para encajar mejor los datos de entrada. Se calculan los pesos al nodo vecino mas cercano para cada nodo, intentando minimizar las distancias entre nodos y sus pesos para hacer que en zonas similares se formen grupos diferenciados.

- *Modular Neural Networks*. Las redes modulares separan elementos o módulos de redes de gran tamaño para tener partes independientes de las cuales se espera un comportamiento específico para conseguir que la combinación de muchos bloques produzca un resultado deseado.

Las redes *Deep Neural Network* o redes de *deep learning* se encuentran implementadas en los siguientes tipos[21,19,14]:

- Modelos No Supervisados y preentrenados. Destacan aquí los *autoencoders* o auto-codificadores, que usan algoritmos de aprendizaje no supervisado. Tratan de buscar representaciones de los datos que permiten hacer una reducción de las dimensiones de los mismos. Desarrollan la idea del análisis de

componentes principales (PCA), con la ventaja de que los *autoencoders* pueden generar representaciones que mantienen relaciones de no linealidad. Utilizan bloques de codificación y decodificación en las capas ocultas para encontrar esas relaciones no lineales. Podría decirse que primero se realiza el análisis a nivel muy detallado de los datos para que después, al alejar la vista, se pueda discernir entre cuáles de dichos detalles son verdaderamente representativos para poder mostrar una representación de los datos de menor peso y similares características. Este grupo incluye las GAN o *Generative Adversarial Networks,* que utiliza ideas de modelos no supervisados para gestionar el uso de dos redes en paralelo. Estas configuraciones permiten generar nuevos datos muy similares a los de entrenamiento. La idea es tener un modelo que genera imágenes similares a un conjunto que se le ha proporcionado para aprender. Seguidamente, se tiene otro modelo que intenta clasificar las imágenes como reales o generadas por el primer modelo. De esta forma, se retienen las imágenes sintéticas que engañan al segundo modelo y son clasificadas como reales para generar (dando nombre así a los modelos generativos) nuevos datos con aspecto similar a los del conjunto dado.

- RBM o *Restricted Boltzmann Machines.* Estas redes, de nuevo no supervisadas intentan encontrar la función de probabilidad que mejor encaja con los datos de entrada. Cada nodo de entrada está conectado a todos los nodos de la capa oculta pero, en una misma capa ningún nodo está conectado entre sí.

- Redes Neuronales Recurrentes. Un ejemplo son las LSTM o *Long Short-Term Memory*. Están diseñadas para ser capaces de almacenar información y de tener control del estado del propio algoritmo. En su estructura cuenta con células a través de las cuales pasa una señal que es alterada por la entrada, una *Forget Gate* o puerta de olvido y la salida. Estos términos son llamados reguladores, que controlan el flujo de información que pasa por la célula. La estructura permite que la célula detecte las dependencias entre la información que le llega en secuencia y a largo plazo.

- Redes Neuronales Recursivas. Son similares a las recurrentes, pueden trabajar con conjuntos de datos de tamaño variable, pero con la diferencia de que son capaces de identificar estructuras jerárquicas en los datos. Un ejemplo es la composición en imágenes, que puede construirse como la serie de objetos que contiene, no solo detectándolos sino aprendiendo las relaciones entre ellos en la escena.

- CNN o Convolutional Neural Networks. Las redes neuronales convolucionales están basadas en el comportamiento del cerebro. Es la de uso mas habitual en la visión por computador. Como es el modelo que se utiliza en este trabajo, se procede a desarrollar con mayor detalle.

4.2. Redes Neuronales Convolucionales CNN

Estas redes tienen la gran capacidad de encontrar objetos entre datos capturados a bajo nivel de abstracción, caracterizados por propiedades que el diseñador del algoritmo basado en CNN no indica expresamente, y basándose en un aprendizaje para el cual cuentan con vectores de información que son representaciones enormemente abstractas. E.g. aprender a detectar un autobús en fotografías que tienen una etiqueta en la que se indica si existe un autobús en ellas o no. En la fotografía, se tienen distintos niveles de intensidad para cada píxel y la red debe encontrar la forma de relacionar estructuras, información redundante, semejanzas de color y de forma entre muchas otras características. A medida que procesa la información de los píxeles, intenta conseguir que ésta información sea cada vez mas compacta y abstracta, hasta poder finalmente decir si en la fotografía hay autobús o no.

En el sencillo ejemplo anterior, debe pensarse que la cantidad de posibles combinaciones de valores de píxeles que una imagen puede tener para que a la vista pueda reconocerse un autobús es infinita (si no se fija un tamaño de imagen). Por esto es que se dice que la red encuentra la mejor representación de los datos, porque la forma que tiene de buscar entre valores y valores es concentrarse en los procesos de abstracción ya que de todos los valores de intensidad de una imagen y todas las posibles combinaciones de valores, la única información referente a la existencia de la clase autobús en la imagen es una etiqueta numérica. Dicha etiqueta numérica es un entero sin signo que debe ser diferente para cada clase que se estima que el sistema aprenda. Entonces para que detecte

autobuses, necesita procesar miles de imágenes hasta que llegue al punto en el que por ejemplo: obvia la información del fondo, porque da igual la escena de la ciudad o el paisaje que atraviesa el vehículo, para esto, se centra en los objetos que tienen neumáticos y para excluir los coches, furgonetas y camiones busca la mayor cantidad de cristales en el objeto, o si nos limitáramos a detectar autobuses londinenses, se centraría en aquellos objetos con neumáticos de gran tamaño y color rojo.

Las interpretaciones *a posteriori* que quien diseñe una red haga son variadas porque ésta actúa como una *caja negra.* Entonces es complejo conseguir afirmar con precisión que la búsqueda que la red realiza se basa en un parámetro concreto pero hay herramientas que ayudan a sacar tal información a la luz. Se intenta utilizar la información que la red pasa de una capa a otra para comprobar el proceso que el algoritmo está siguiendo durante el procesado.

Hecha esta pequeña introducción, puede pasarse a analizar la estructura de las CNN con algo mas de detalle.

4.2.1 Estructura

Como se ha mencionado, las redes de *deep learning* suelen tener una estructura de capas compuestas de nodos. Las CNN cumplen esta idea. Dentro de esa arquitectura, las CNN realizan ciertas funciones que pasan a detallarse:

- Convolución
- Muestreo
- Operaciones de detección/clasificación
- Normalización

4.2.2 Tipos de capas

4.2.2.1 Capas de Convolución

Las capas de convoluciones son capaces de realizar una extracción de características cada vez mejor y de mayor detalle. Se basan en la operación de convolución, que se trata de una operación matemática en la que dos señales (que pueden ser la misma), se van multiplicando y desplazando a la vez que se mantiene el registro de la acumulación de esos valores. La operación tiene la siguiente forma en su versión para señales continuas:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

Y en su versión discreta:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Y su versión con un filtro *w* de dimensiones finitas con una altura H y un ancho W, como se implementaría en el uso de imágenes:

$$(f * w)[m,n] = \sum_{j=-W/2}^{W/2} \sum_{i=-H/2}^{H/2} f[m,n]w[m+i,n+j]$$

Como puede verse, una señal es invertida y va trasladándose en el tiempo a medida que se multiplica por la otra señal. El resultado de integrar el proceso anterior da una salida alta en aquellos instantes en los que las señales también tienen alto valor y una salida pequeña cuando las señales presentan valores bajos. Así, si se realiza la convolución de una señal con otra, se tendrán como máximos, los puntos localizados en las zonas que coincidan con altos valores en ambas señales. Así se puede tener una medida aproximada de la similitud entre señales, cuestión que resulta eficazmente aprovechada a modo de filtros de convolución por las redes neuronales.

En las CNN se utiliza un número de filtros de convolución de un tamaño concreto en cada capa. En las fórmulas expresadas anteriormente se mostró la convolución en una dimensión, en las imágenes que son objeto del procesado por redes CNN, se utilizan convoluciones en dos dimensiones, que no son mas que un desarrollo del mismo concepto. En las imágenes es habitual utilizar una máscara de convolución, con la cual se selecciona una zona de una de las imágenes para que sea procesada con otra imagen completa. Al aumentar o disminuir el tamaño de dichas máscaras o núcleos, se pueden encontrar coincidencias de patrones de mayor o menor tamaño, que presenten características localizadas en los diferentes tamaños que pueden ser seleccionados para realizar las mencionadas operaciones.

Los nodos o neuronas de una capa convolucional no están conectados a todas las neuronas de las capas adyacentes. Únicamente se conectan a los puntos que se solapan ligeramente con el subconjunto actual y esta división en regiones produce los llamados campos receptivos locales o *local receptive field.* Las neuronas en un mismo filtro o campo concreto trabajan buscando las mismas características pero en diferentes áreas de la imagen, de forma que pueden compartir valores de pesos para aligerar el comportamiento y la eficacia.

Es habitual agregar tras la capa de convolución una función que aplique una no linealidad, como por ejemplo, una ReLU justo a la salida.

4.2.2.2 Capas de muestreo: subsampling/pooling

Tras señalas las zonas relevantes en las capas de convolución, es habitual reducir la cantidad de información generada, para así también limitar las posibles características que se van a encontrar, de forma que la red tenga que reajustarse para limitar la información que mejor representa y resuelve los objetivos.

De esta forma, se reduce el tamaño de la red. Así se hace a la red también bastante sólida ante la detección de giros y traslaciones de los objetos.

El subsampling o submuestreo se consigue realizando operaciones de *pooling*. Se utiliza porque los mapas de características que las capas de convolución generan son muy susceptibles a la posición concreta en la que se dan las ocurrencias. Así, pequeñas diferencias en la imagen generan mapas de características diferentes. Al realizar el muestreo, se retiene la información de la imagen sin los detalles finos y precisos, con la idea de que los aspectos de mayor peso o estructurales permitan resolver la tarea.

La operación de pooling se realiza sobre una máscara o mapa de características, es decir, de forma localizada. La forma de operar consiste en seleccionar áreas que no se superpongan y filtrar eligiendo de los valores del área uno según dos criterios principalmente: filtrado eligiendo la media de los valores o filtrado eligiendo el valor máximo de los que existían en el área. Esto produce un muestreo no lineal que como se ha dicho, hace a la red sólida ante diferencias de tamaños pequeños entre objetos o características dentro de las imágenes. (Ver figura 10).

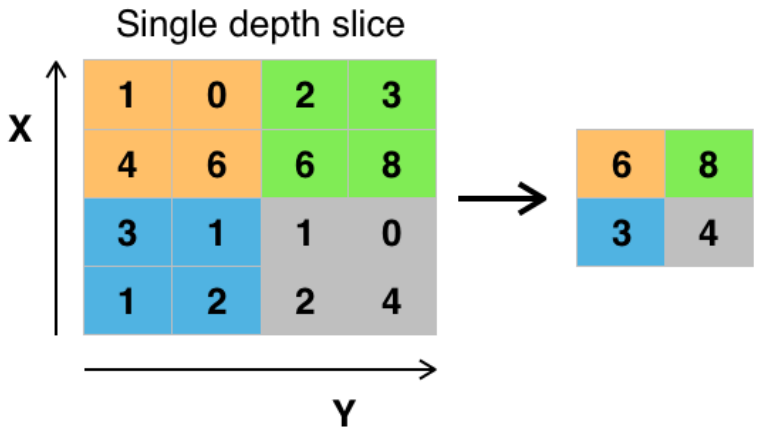


Figura 10: Operación de MaxPooling. Fuente: Wikipedia, https://upload.wikimedia.org/wikipedia/commons/e/e9/Max_pooling.png

4.2.2.3 Capas de detección/clasificación: capas densas o fully conected

Las capas densas, agregadas al modelo consiguen realizar la tarea de clasificación. Así, internamente, los campos receptivos locales o mapas de características que se generan en las capas de convolución son clasificados y agrupados en torno a características conforme a la evolución de los pesos de estas capas. Este proceso va realizándose de forma repetida y a diferentes niveles de abstracción hasta que finalmente se es capaz de resolver la predicción.

Las capas de este tipo siguen la estructura reflejada en el inicio de este capítulo.

4.2.2.4 Normalización en las capas

Las capas que realizan normalización son aquellas en las que se comprimen los valores de entrada en rangos normalizados. Esto permite que posteriormente puedan interpretarse cómodamente los datos en los siguientes pasos del proceso.

La técnica de normalización que actualmente se impone es el *BatchNormalization*. Antes de pasar por una explicación de en qué consiste, debe concretarse el detalle de qué es un batch. En los algoritmos de *machine learning*, es habitual trabajar con grandes cantidades de datos, las cuales son normalmente procesadas muchas veces por una red hasta que consigue generalizar una solución (si lo consigue). Cuando se realiza un entrenamiento, cada vuelta entera que se da a todo el conjunto de datos se llama una ***epoch***. Dentro de cada *epoch*, es habitual tomar grupos de datos de pequeña cantidad llamados ***batch***. Éste último conjunto se refiere a la cantidad de muestras individuales que se desea sean procesadas por la red antes de calcular los nuevos pesos. Éste es un dato de interés en el diseño de redes y en su entrenamiento, ya que debe controlarse en función de la variedad de datos disponibles, así como la disparidad entre ellos.

La normalización de *mini-batch* o de un mini-lote toma la media de cada característica y la sustrae de las muestras además de dividirla entre la desviación típica. Esto permite que las muestras tengan homogeneidad y controla la variación en los valores de entrada de la red, mejorando la capacidad de optimización de esta al presentar menos situaciones en las que los datos parecen muy diferentes y sin embargo representan una situación similar, generando desviaciones o saltos en la dirección de mejora de la red. Así se reduce la posible tendencia a dar saltos en el espacio de soluciones sin mantenerse fijo en la dirección de la solución óptima.

Existen otras posibilidades con la normalización. Es posible normalizar los pesos de las conexiones entre los nodos de la red dividiendo cada vector w en su magnitud y dirección. También se normalizan las capas al normalizar cada entrada individualmente. Hay otras posibilidades en cuanto a normalización, con capacidades muy especializadas que pueden resultar de provecho, como: la normalización de grupos, de instancias, la re-normalización de lote...

4.3. Algoritmos de aprendizaje

Los modelos siguen las pautas dictadas por un algoritmo de aprendizaje para conseguir optimizar su funcionamiento conforme a la tarea solicitada. Estos son una evolución bio-inspirada de la optimización lineal y los métodos de regresión lineales. El objetivo final es encontrar los valores de los pesos en las conexiones entre nodos que mejor resuelven los problemas dentro de un dominio.

4.3.1 Descenso de Gradiente (GD)

El descenso de gradiente toma la idea del método de Newton. En una función en 2 dimensiones se puede elegir un punto de forma aleatoria y moverlo a la izquierda y a la derecha en función del signo de la derivada que en una u otra dirección la función plantea. Continuando el proceso se puede llegar a las posiciones en las que el valor de la función se minimiza. En el caso de funciones n-dimensionales el problema puede escalarse para buscar en todas las direcciones del espacio en el que existen.

Algunos problemas de éste método son: dificultades para escapar de mínimos locales, complicaciones con el uso de funciones no convexas,

4.3.2 Descenso de Gradiente Estocástico (SGD).

Es el método mas utilizado en el entrenamiento de algoritmos de *deep learning*. En el descenso de gradiente se actualizan los valores de los pesos tras procesar todas las muestras del conjunto de entrenamiento. En el SGD se actualizan los pesos tras un lote de *n* muestras. Como el SGD varía los pesos mucho mas rápido que el GD, se converge hacia el mínimo de forma mucho mas rápida.

Esta selección del lote se realiza de forma aleatoria, y ofrece una mayor cantidad de iteraciones que el GD. Intenta aproximar el objetivo en pequeños pasos y puede converger tras recorrer los datos varias veces.

4.3.3 Momentum

En el SGD, cuando se emplea el método estándar se utiliza un parámetro llamado ratio de aprendizaje que permanece fijo como un valor multiplicado al gradiente. De esta forma se puede controlar el tamaño de cada paso que se da al actualizar los pesos en la dirección que minimiza las pérdidas. De esta forma, podría pasarse por encima de un mínimo local si el gradiente tiene un salto demasiado grande o se podría tardar mucho en converger si el paso es demasiado pequeño.

El concepto de momento de inercia de la física puede aplicase al problema de aprendizaje en *deep learning*. Se utiliza una variable de velocidad que es configurada como una exponencial negativa que actúa sobre la media del valor del gradiente, lo que ayuda a prevenir recorridos en direcciones erróneas.

4.3.4 Dificultades en los algoritmos de aprendizaje

Algunas complicaciones habitualmente encontradas durante los entrenamientos de redes neuronales artificiales con los métodos mencionados son:

- Overfit: como se ha descrito, el sobreajuste es provocado por el uso de un modelo de gran capacidad y un entrenamiento exhaustivo en un conjunto de datos limitado. Así el modelo es capaz de "memorizar" los datos y conseguir una precisión absoluta.

- Convergencia temprana: al contrario que el sobreajuste, es posible tener situaciones en las que un modelo de capacidad insuficiente para la tarea y es incapaz de mejorar, llegado a un punto, su funcionamiento durante el entrenamiento.

- Desvanecimiento de gradiente: con el uso de funciones sigmoides, es posible que un gradiente se encuentre reduciendo su valor de forma continua al navegar por la función en el sentido negativo del eje de abcisas.

- Sobrecrecimiento de gradiente: como cada parámetro que es variado genera una cascada de variaciones en todas aquellas variables que dependen de la primera, es posible a veces que cuando ciertos gradientes crecen desmesuradamente se cree un efecto dominó que hace que los gradientes "exploten".

- Mínimos locales: en el caso de funciones convexas, alcanzar un mínimo local conlleva alcanzar el mínimo global. En el caso de funciones no convexas, es un problema conseguir que los algoritmos de aprendizaje sean capaces de escapar de esas zonas para seguir buscando el punto de mínimo global.

- Regiones planas: los algoritmos basados en descenso de gradiente también encuentran grandes dificultades en regiones planas de funciones no convexas, pudiendo incluso llegar a parar el algoritmo.

- Tiempos de entrenamiento: los algoritmos de aprendizaje en *deep learning* lidian con millones de parámetros y miles de muestras de entrenamiento. Esto provoca que en ocasiones en las que se buscan características poco evidentes entre muchísimos datos se tengan tiempos de entrenamiento muy largos.

4.3.5 Hiperparámetros

Los hiperparámetros mas utilizados en el *deep learning* son el ratio de aprendizaje y los parámetros de regularización. Las CNN tienen otros: número de filtros, tamaño de la máscara de cada filtro...

Como el ratio de aprendizaje tiene un gran impacto en los procesos de entrenamiento, existen múltiples herramientas que ayudan a controlarlo y mejorar el comportamiento para recorrer con el mayor acierto posible zonas planas, zonas con grandes caídas, funciones no convexas… Los métodos de adaptación del ratio de aprendizaje mas habituales son:

- Delta bar: Aumenta el ratio de aprendizaje si la derivada parcial respecto del ratio de aprendizaje de cada peso mantiene el signo. Si el signo cambia, reduce el ratio.

- AdaGrad: AdaGrad adapta el ratio de aprendizaje en función de la frecuencia con la que se dan ciertas características. Reduce el ratio de aquellas características que son muy frecuentes y aumenta el ratio de aprendizaje para aquellas características que son menos comunes. Evita tener que afinar manualmente el ratio de aprendizaje. Su desventaja es que termina por hacer cero algunos gradientes ya que el ratio de aprendizaje se divide entre la raíz cuadrada de la suma al cuadrado de los gradientes anteriores, pudiendo entonces hacer el ratio casi cero. Esto se intenta resolver en otro algoritmo llamado AdaDelta, de funcionamiento similar.

- RMSProp: éste método divide el ratio de aprendizaje entre la media de los gradientes pasados al cuadrado, que decae exponencialmente. Como AdaDelta, trata de resolver el problema de AdaGrad con el desvanecimiento del ratio.

- Adam: es el método de uso mas habitual hoy en día. Su nombre surge de *Adaptive Moment Estimation*. También guarda una serie de medias de gradientes que decaen exponencialmente como RMSprop o AdaDelta, pero incluye otra serie de medias de gradientes similares al momento de inercia.

4.3.6 Backpropagation

El algoritmo de *backpropagation* o "propagación hacia atrás" tiene un funcionamiento sencillo. Primero, la entrada es procesada en dirección a la salida, por toda la red. Tras este paso, se calcula el error entre esa primera predicción y la verdad de referencia. Después, el error se propaga hacia atrás mientras se ajustan los valores de los pesos para reducir el valor del error.

Primero se calcula el error en cada neurona e_k entre el valor deseado d_k y el valor obtenido y_k:

$$\delta_k = e_k g'(y_k)$$
$$e_k = d_k - y_k$$

Ese error se propaga hacia atrás y se va calculando su producto con la derivada de la función de activación *g'(·)*.Esa cualidad de derivabilidad de las funciones de activación mencionada anteriormente se ve aquí ahora justificada. Cuando se calculan los valores delta, se buscan los de las capas anteriores en función de la capa actual:

$$\delta_j = \eta g'(y_j) \sum_{k=0}^{K} \delta^k w_{jk}$$

Donde la letra griega eta representa el **ratio de aprendizaje**, que modela la variación en los pesos que va a

calcularse. Finalmente se puede calcular el incremento en cada peso, en cada conexión para continuar con el algoritmo tras las siguientes entradas y ajustando de nuevo mas valores en función de las salidas:

$$\Delta w_{jk} = \delta_j y_k$$

Este proceso se repite hasta alcanzar un criterio de parada, normalmente basado en el cálculo del error, por ejemplo, una parada programada al llegar a un punto de estancamiento o convergencia en los intentos de reducción del error. En la primera iteración del algoritmo se tienen diversas opciones: iniciar los pesos de forma aleatoria, cargar los pesos de una red preentrenada...

5 MÉTODO PROPUESTO

El proyecto que aquí se describe, como se ha mencionado está dentro del marco de la realización de unas prácticas en FISEVI, trabajando en el proyecto POMPIS. El proyecto tiene como objetivo primario el desarrollo de una aplicación de Inteligencia Artificial (AI) en red neuronal que permita predecir áreas de displasia de alto grado en canal anal, mediante el aprendizaje y análisis de imágenes tomadas con Anoscopia de Alta Resolución (HRA), mejorando la rentabilidad a la toma de biopsias.

El proyecto POMPIS se estructura en cinco fases:

1. Adquisición y normalización de imágenes

2. Procesamiento de las imágenes almacenadas. Orto-macrofotografías

3. Diseño del enfoque de Aprendizaje automatizado

4. Diseño del enfoque de Redes neuronales para clasificar las imágenes

5. Análisis estadístico de los diferentes patrones de imágenes y su relación con HSIL.

La primera fase se desarrolla desde los años 2011-2012 y tiene lugar en el HUVR. Se han tomado de forma sistemática fotografías y vídeos del desarrollo de la enfermedad durante las intervenciones clínicas a los pacientes. Se dispone de una base de datos de más de 8000 imágenes y vídeos de HRA. Las imágenes son almacenadas en soportes digitales. En el nombre de cada archivo de imagen se ha añadido: un identificador o Número de Historia Clínica, que es una referencia que utiliza cada centro hospitalario para designar unívocamente a un paciente; la fecha en la que las imágenes fueron tomadas y una predicción del médico que tomó la imagen acerca del grado o el estado de la enfermedad. Durante el proceso, los médicos además describen la intervención clínica en una hoja de evolución. Algunas veces, si encuentran patrones que resulten sintomáticos, pueden tomar una o varias biopsias y realizar citologías, así que también existe una hoja de resultados patológicos de aquellos pacientes a los que se han tomado muestras. Entonces, de forma general, para un paciente se tienen datos de diversas intervenciones, en forma de imágenes y texto, describiendo la intervención y los resultados.

La segunda fase ocupa una buena parte del trabajo realizado. Las imágenes no han sido ordenadas ni debidamente catalogadas. Los datos disponibles son fotografías realizadas durante las sesiones, tomadas por los médicos y almacenadas posteriormente. Existe entonces la necesidad de generar álbumes en los que se tengan disponibles las imágenes relativas a un paciente, y a una fecha de intervención, o también, la capacidad de agrupar todas las fotos en las que se haya diagnosticado un cierto grado de displasia. Estas funcionalidades son realizadas junto con otras, como la traducción de NHC a NUHSA, creando para ello referencias para los pacientes, con las que podemos pasar cómodamente de un identificador a otro, y atacando el problema en pequeñas tareas por medio del uso de scripts, o pequeños bloques de código de programación.

La tercera fase se engloba con la cuarta, para simplificar, en este proyecto. Para la comprensión de los sistemas basados en Redes Neuronales se realiza un estudio teórico del tema, que queda desarrollado en el capítulo inicial de esta memoria. Además, se realizan diversos ejemplos para afianzar el entendimiento de las herramientas y comprender sus posibilidades prácticas. Se prueban con clasificadores de imágenes sencillos, redes convolucionales para la segmentación de imágenes y clasificadores de mayor complejidad.

La quinta fase tiene un desarrollo que inicia cerca del final de la tercera y la cuarta. Quiere esto decir que está limitado su inicio al momento en el que las fases anteriores comiencen a dar resultados. El análisis estadístico de los patrones será lo que permita dilucidar cuáles son mas significativos para el desarrollo de la enfermedad y en cuáles los médicos han de prestar mayor atención.

Así, la propuesta general será la de realizar una aplicación basada en redes neuronales que sea capaz de clasificar imágenes entre los distintos grados de displasia que los tejidos pueden presentar. La estructura es de una red VGG16. Además, se hace un programa con interfaz gráfica para realizar la extracción de patrones en las imágenes, que coteje la información de las Hojas de Evolución que los médicos escriben tras las intervenciones y los resultados de las biopsias que los patólogos realicen, para recortar aquello que los profesionales dicten como verdad de referencia. Con dicho programa se extraen los datos para entrenar la red, una vez la base de datos de imágenes queda organizada. Y finalmente se prueba la red con datos extraídos con la aplicación para comprobar como evoluciona la capacidad de la misma.

5.1. Herramientas y situación de la base de datos

Las herramientas que se disponen son variadas. Para la programación general de las funciones y aplicaciones, así como el entrenamiento de la red y la gestión de la base de datos se cuenta con un portátil MSI modelo CX62 6QD con Microsoft Windows 10.

La base de datos consta de unas 7500 imágenes de HRA almacenadas en formato físico, pero que termina alojándose en un servidor unido a la red corporativa del Hospital Universitario Virgen del Rocío. Las imágenes, como se ha comentado, son capturadas y luego almacenadas por los médicos. Éstos realizan el cambio del nombre de archivo después de cada sesión clínica. Este es un proceso que sería deseable automatizar, permitiendo al médico seleccionar diversas imágenes y aplicarles a todas un nombre automáticamente que incluya la existencia de enfermedad, la fecha de intervención, el número de imagen de esa intervención y el identificador de paciente.

Las imágenes son capturadas con varios dispositivos a lo largo de los años, pero tienen en común el ser cámaras fotográficas de alta calidad. Un gran número de imágenes se toma con una Olympus EP-3, otro gran grupo es capturado con una Olympus E-M10Mark II. Así, en el momento de recibir los datos, se dispone de tres carpetas llenas de imágenes. Cada una contiene las imágenes relativas a pacientes cuyo identificador NHC comienza por cien, por doscientos o por trescientos mil. Según aumenta el ID, la cantidad de fotos se reduce, así que en la carpeta 100m1 hay contenidas más imágenes que en la 200m1 y en ésta, más que en la carpeta 300m1. Cada fotografía tiene entonces: un identificador, una fecha y separado con guiones bajos, '_' observaciones sobre la situación de la enfermedad así como una letra por cada captura de una misma sesión, es decir, se añade al final del nombre del archivo un '_a', '_b', '_c'… y así sucesivamente. Todas las capturas están en formato JPEG.

Casi todas las imágenes están en alta resolución, de al menos 4k, es decir, 4608 píxeles de ancho y 3456 píxeles de alto. La profundidad de bit es de 24. El espacio de color es en la mayoría de imágenes el sRGB, y la captura se realiza con ajustes de ISO y enfoque automáticos.

Como la toma de imágenes se extiende durante unos seis o siete años, es lógico que los profesionales actualicen los equipos, originando inconsistencias en la base de datos que deben tenerse en cuenta para evitar futuras limitaciones. La cuestión que nos afecta es la diferencia de resolución de las imágenes, ya que hay algunas de 3056 x 2292píxeles, otras de 4032 x 3024 y de 3717 x 2788. Esto, como se verá, debe tenerse en cuenta durante la extracción de los recortes de los patrones.

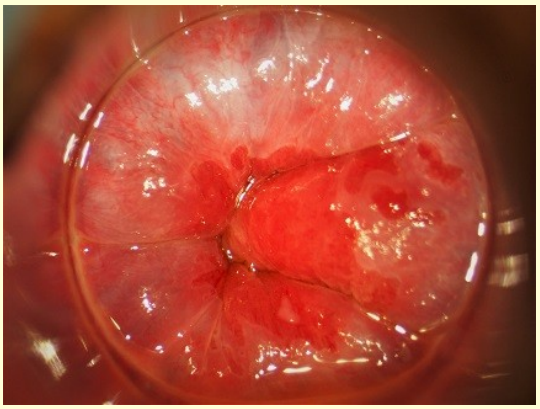
Debido a que todo el proyecto conlleva funcionalidades muy variadas, se apuesta por el entorno **Anaconda**, para facilitar el uso como lenguaje de programación de **Python**. Python es un lenguaje multiparadigma de uso general que resulta cómodo al permitir trabajar con scripts o pequeños bloques de código, resolviendo tareas diversas con enfoque similar.

Python pone a disposición del usuario la utilización de librerías que contienen funciones de alto nivel para trabajar en muchas áreas.. Está especialmente instaurado entre la comunidad científica, siendo Anaconda una herramienta

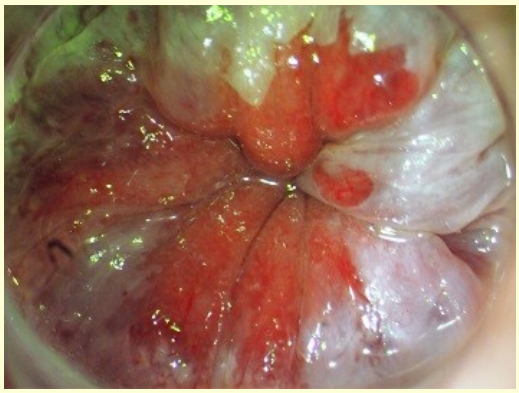
popular para controlar y gestionar todas las librerías. Se utilizan muchas, las más importantes son: **numpy**, que ofrece la posibilidad de crear variables de tipo matriz y vector, así como muchas operaciones y funciones matemáticas y control de alto nivel sobre las matrices y vectores; **OpenCV**, es una librería de visión artificial, con la que se trabaja cómodamente con las imágenes, ofrece diversas funciones como transformadas, control del color, operaciones morfológicas, reconocimiento de objetos… es una librería muy utilizada actualmente en visión por computador; **Keras** es la librería elegida para generar la estructura de red neuronal, hay actualmente muchas otras, Keras ofrece una gran comunidad muy activa que resuelve problemas conjuntamente desde hace años, no tiene una documentación demasiado extensa pero resulta suficiente; **TensorFlow** es una librería que permite el uso eficaz de estructuras vectoriales o matriciales que las redes neuronales necesitan; **matplotlib** es una librería que permite generar gráficos y exponer resultados con relativa sencillez, admite muchas opciones a la hora de realizar la representación de datos; además se utiliza **os**, que permite el uso de primitivas del sistema operativo, muy útil para controlar el flujo de datos desde los directorios a los scripts. Estas son las librerías que la comunidad utiliza, así que por cuestiones de homogeneidad son elegidas.

La tabla 3 tiene una muestra que pretende ilustrar la variedad de imágenes que contiene la base de datos:

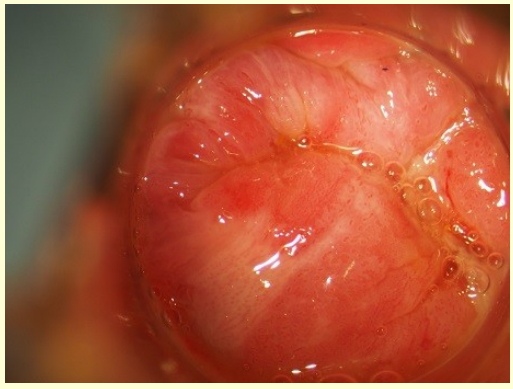
a)



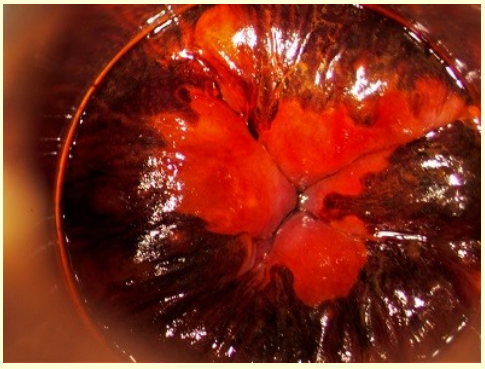
b)



c)



d)



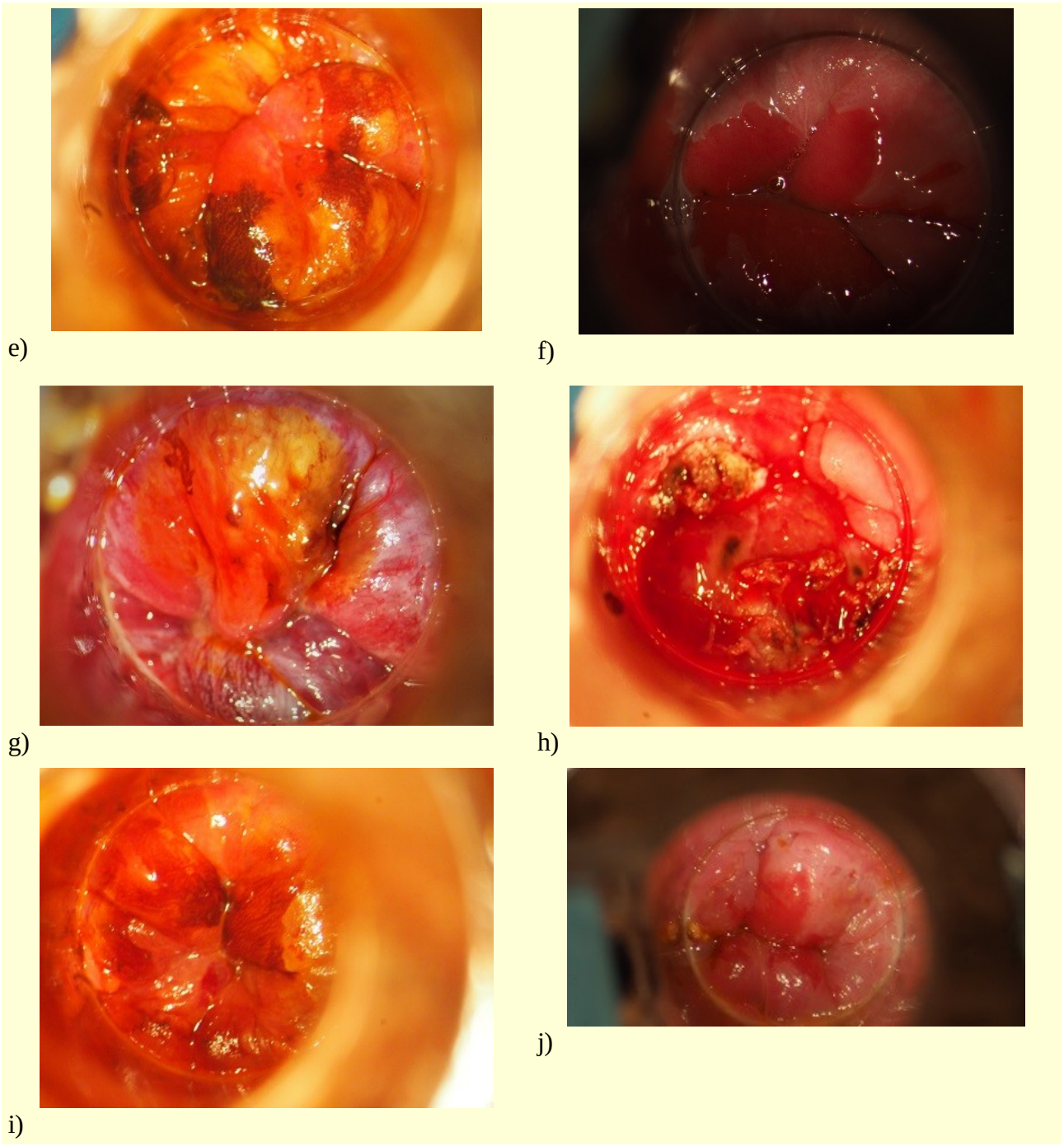


Tabla 3: Detalle de las diferentes imágenes encontradas en la base de datos. De izquierda a derecha y de arriba hacia abajo: a) imagen normal, b) imagen con ácido acético muy visible, c) imagen con ácido acético poco visible, d) imagen con Lugol con respuesta positiva, e) imagen con Lugol con respuesta negativa, f) imagen con baja iluminación, g) imagen con Lugol sin aplicar en toda la zona, h) imagen tras termocoagulación, i) imagen con la zona de interés descentrada., j) imagen con excesivo recorrido del anoscopio

Se tiene entonces una base de datos con imágenes sin etiquetar, de características diversas, sin un almacenaje concreto y ordenado. Todo esto no supone error alguno, puesto que se guardan las fotografías para su estudio posterior, y la realización de una aplicación que aprenda a clasificarlas es simplemente otra forma de aprovechar tanta información. Pero eso no quita que para utilizar como herramienta las técnicas de *Deep Learning* es necesario preparar exhaustivamente los datos, han de quedar normalizados, ajustados en tamaño, etiquetados en función de las clases a las que pertenezcan, deben extraerse las verdades de referencia y discriminar los datos que son menos relevantes es también una necesidad. Por esto, la preparación y adecuación de las imágenes representa un objetivo más que digno y que se sigue describiendo en las secciones siguientes.

5.2. Entornos de entrenamiento en deep learning

Es habitual en tareas relacionadas con la programación que se trabaje con entornos creados para que resulten cómodas ciertas cuestiones ya que la especialización hace que incluso en proyectos muy distintos que usan tecnologías similares se tengan en común ciertas necesidades técnicas. Tanto es así que por ejemplo, para la programación general existen muchas aplicaciones y programas que aglutinan muchas funcionalidades con aras a facilitar el trabajo.

Existen muchos ejemplos de entornos o *frameworks* dedicados. Por ejemplo para el uso de varios lenguajes de programación como Java o Python existe el entorno Eclipse IDE. Eclipse pone a disposición un editor que responde al lenguaje en el que se codifica, una amplia batería de intérpretes y compiladores, funcionalidades para el manejo de árboles de directorios y muchas otras herramientas. Otro ejemplo muy utilizado es Visual Studio, común entre programadores de C,C++,R... y que también tiene asistentes en la edición, en la navegación, herramientas para hacer depuración de código etcétera. Podrían nombrarse muchas otras herramientas como GNU Emacs, soluciones como Notepad++ o Sublime Text.

El caso del tratamiento de datos y el *deep learning* no es distinto de los mencionados. Debido a que mas allá de la técnica se ha programar, sea *deep learning* o cualquier otra, en un entorno que resulte cómodo y esté mínimamente adaptado al lenguaje, por cuestiones por ejemplo de indentación. En ese sentido para ciertas tareas en el proyecto se ha utilizado Notepad++ como editor de Python. Notepad++ es una herramienta bastante útil que además de ser gratuita es de código abierto. Además de aceptar Python, permite editar ficheros de texto grandes con comodidad ya que puede buscar ocurrencias de una cadena de caracteres y reemplazarlos o reemplazar las líneas en las que exista esa cadena, o eliminar todas las líneas que no contengan la cadena…

Con los ficheros de Python .py escritos en el editor se pueden ejecutar scripts desde el *shell* o la línea de comandos del sistema operativo, pero para ello es necesario instalar el intérprete de Python y todas las librerías que sean usadas. Como es habitual necesitar múltiples librerías es aconsejable utilizar un interprete de Python integrado en un entorno que permita gestionarlo y manejar las librerías. Para esto es utilizado Anaconda, ya que es el más común entre la comunidad y tiene a disposición una documentación profunda. Entonces una vez instalado Anaconda hay que ir agregando las librerías a un espacio de trabajo accesible desde la línea de comandos que podrá ejecutar scripts de Python cómodamente y consumiendo mínimos recursos.

Sin embargo, la excesiva austeridad del método anterior conlleva que resulte poco útil para trabajar con programas o scripts de los que sea deseable observar resultados gráficamente ni tampoco trabajando con pequeños bloques de código que requieren de fases de edición-ejecución-depuración muy cortas. Debido a que tal es el caso de la programación con técnicas de *deep learning* se utiliza Jupyter en el trabajo del cual se comenta su uso a continuación.

De forma poco llamativa y bajo Python, Jupyter y Anaconda se tiene la interfaz TensorFlow. Es la herramienta que permite ejecutar algoritmos de muchos tipos, sobre todo de machine learning, en una gran cantidad de dispositivos diferentes sin apenas realizar cambios en el código, teniendo a disposición dispositivos móviles, tablets o sistemas distribuidos en cientos de máquinas, o clústeres de unidades de procesamiento gráfico (GPU) e incluso unidades de procesamiento tensorial (TPU). Es la herramienta que trabaja permitiendo expresar con eficacia algoritmos de entrenamiento para modelos de redes de deep learning. TensorFlow está creado y tutelado por Google y ha sido utilizado para que sistemas puedan desarrollar: mejoras en las búsquedas del motor de Google, reconocimiento de voz, Google Maps y StreetView, Youtube y muchos otros. Una alternativa muy utilizada es Theano, que también es de código abierto y pone a disposición herramientas similares a las de TensorFlow, llegando incluso a ser en ocasiones más adaptable si se sabe configurar bien, pero es precisamente este detalle el que hace que en este trabajo se opte por usar TensorFlow, ya que apenas requiere configuración alguna, incluso puede accederse a una versión de Keras que queda dentro de TensorFlow, con lo que las incompatibilidades o dificultades deberían quedar reducidas.

5.2.1 Uso de Anaconda y Jupyter

El primer entorno de pruebas de entrenamiento se compone de Anaconda y Jupyter. A falta de una computadora especializada, se realiza con el equipo personal descrito al inicio del método propuesto.

Jupyter es una herramienta para programar desde el navegador web. Utiliza una dinámica servidor-cliente, en la que existe la posibilidad de trabajar de forma remota o local sin que ello conlleve grandes diferencias para el funcionamiento final de la aplicación es decir, el usuario únicamente conectando su computador a otra máquina puede abrir su cuaderno o entorno y trabajar de forma remota. Además hay varias plataformas de servicios de alquiler de harware en la nube compatibles con Jupyter como Amazon Web Services, Google Cloud y Google Colab, Paperspace Gradientº y más.

Trabaja con ficheros .jnpy que guardan en células o bloques la información escrita. Jupyter distingue varios tipos de células en las que escribir, por ejemplo, de texto y de código. La idea es que aquellos bloques en los que el usuario puede escribir puedan servir para introducir y explicar lo que se desea lograr en los segmentos de código. Pero no se limita a texto plano sino que es posible crear un documento formal en el que se explique todo el funcionamiento de un programa junto con el código del mismo y los resultados bloque a bloque de su ejecución. Y en las células de código se pueden tener pequeñas aplicaciones o funciones para ejecutalas independientemente. En un cuaderno, las variables que sean creadas en un bloque de código pueden ser usadas en otro, y las funciones son también accesibles. También, ofrece la capacidad de separar los resultados de cada bloque que incluso quedan guardados y son visibles al cerrar el cuaderno y volver a trabajar en él. Todo esto lo convierte en un editor en tiempo real en el que puede ejecutarse un programa mientras se escribe.

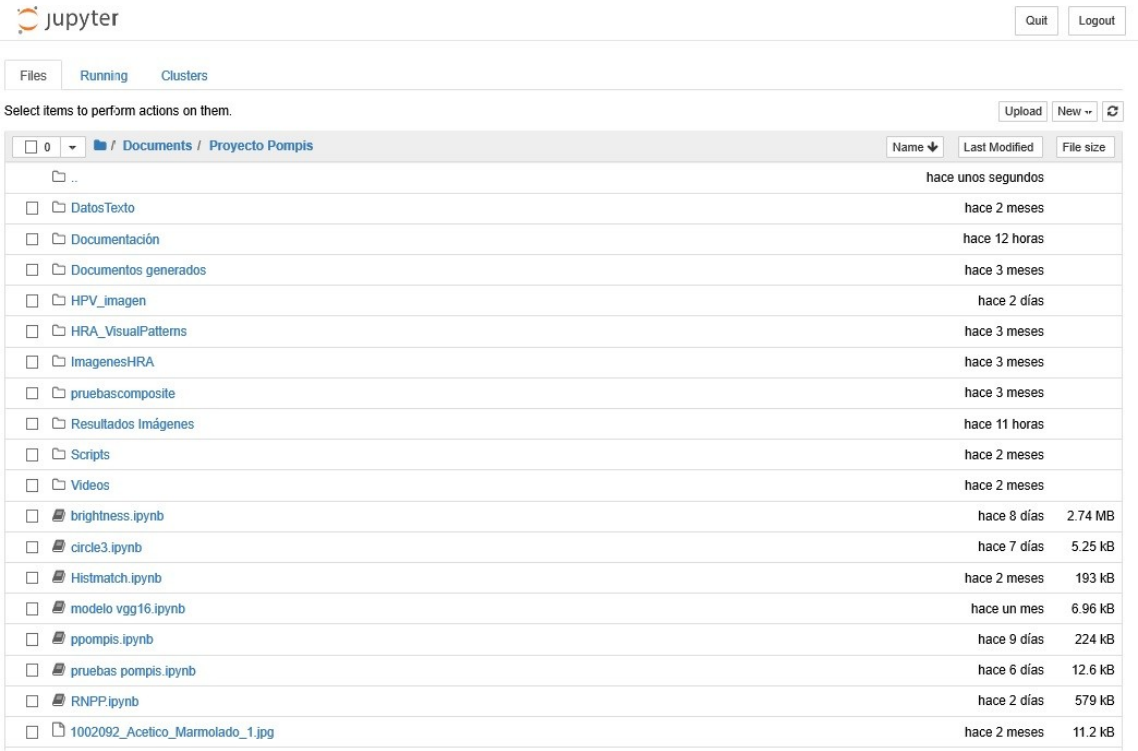


Figura 11: Aspecto del menú de Jupyter

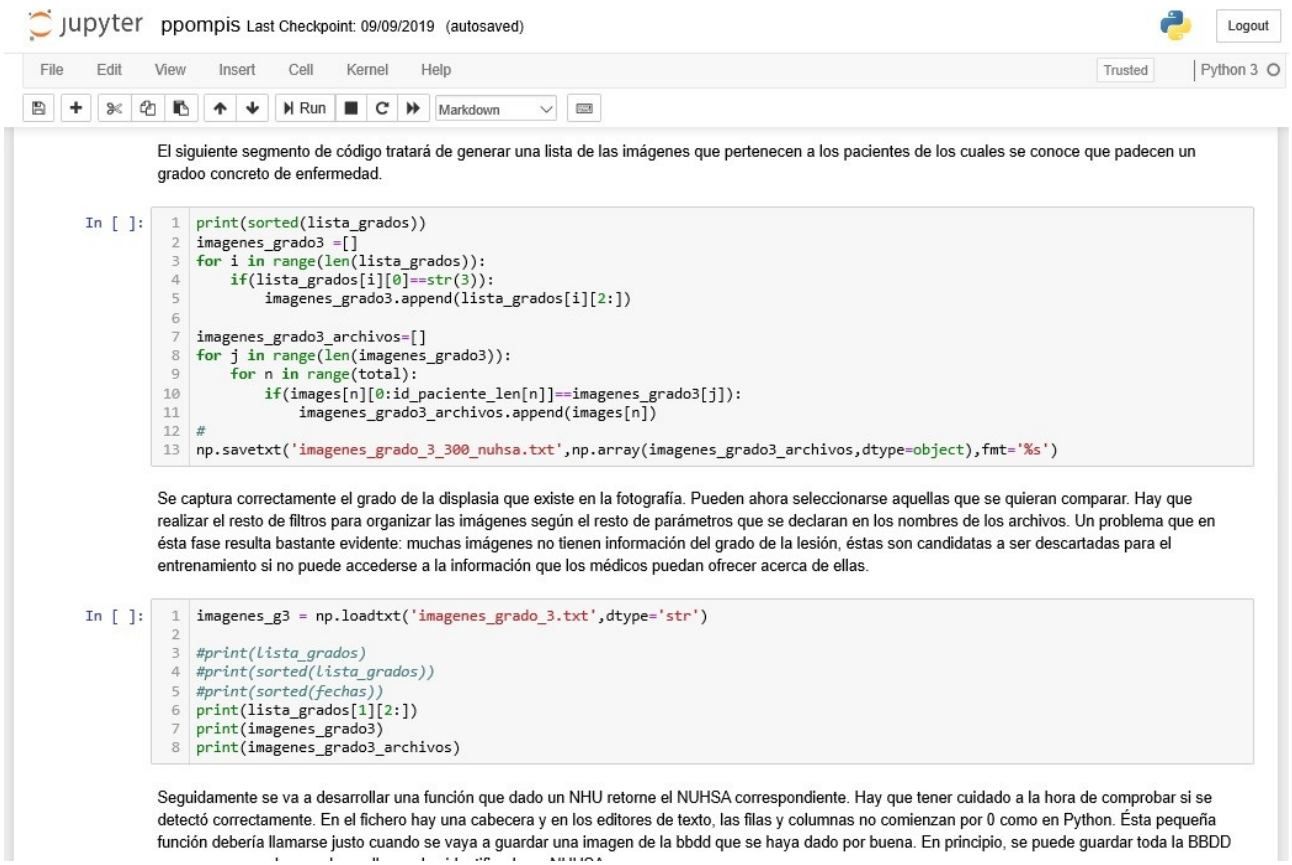


Figura 12: Aspecto de un cuaderno Jupyter

El uso de Jupyter se limita al entorno local de la computadora descrita al inicio del método propuesto. El trabajr de forma local conlleva algunos pasos y tiempos de espera para el inicio de la aplicación que en programas con menor orientación al uso remoto y a la edición de documentos no se tendrían. Esto sin embargo apenas resulta un lastre porque el funcionamiento de Jupyter resulta fluido. La gran ventaja de usar Jupyer incluso en una máquina poco conveniente para tratar y procesar grandes cantidades de datos es que los cuadernos son fáciles de reutilizar en otra máquina o como se ha comentado, de forma remota. De todas formas, con Python se tienen muchas comodidades a la hora de por ejemplo hacer el código independiente del sistema operativo, con el uso de librerías como os, que por ejemplo permite obviar las diferencias en la forma de gestionar rutas de directorios entre un sistema y otro, con lo cual se gana también la posibilidad de con poca edición correr el código en otra máquina con un sistema diferente del usado.

5.2.2 Uso de Jupyter con Google Colaboratory

Colaboratory es un entorno gratuito de Jupyter Notebook que es ejecutado en la nube. Brinda acceso a recursos informáticos y computacionales de gran potencia.

Google Colaboratory está integrado con la plataforma de almacenamiento Google Drive. De esta forma se puede tener un conjunto de datos en la nube para cargarlo desde el cuaderno Jupyter de forma similar a como se hace trabajando de forma local, pero evitando tener que ocupar en exceso la conexión a internet. Todas estas comodidades hacen posible el entrenamiento de redes neuronales sin el hardware adecuado, allanando el camino y democratizando el proceso. En la figura 13 puede verse el aspecto de Google Colab. En el centro se tiene el cuaderno de Jupyter tal y como se presenta en su uso directo con Anaconda. A la izquierda se tiene un menú con la estructura de archivos e índices para acceder de forma rápida a secciones concretas del documento. En la figura mencionada puede observarse a la izquierda en el menú la estructura de los archivos, con acceso a Google Drive desde la máquina virtual, que por su estructura de ficheros es basada en GNU/Linux.

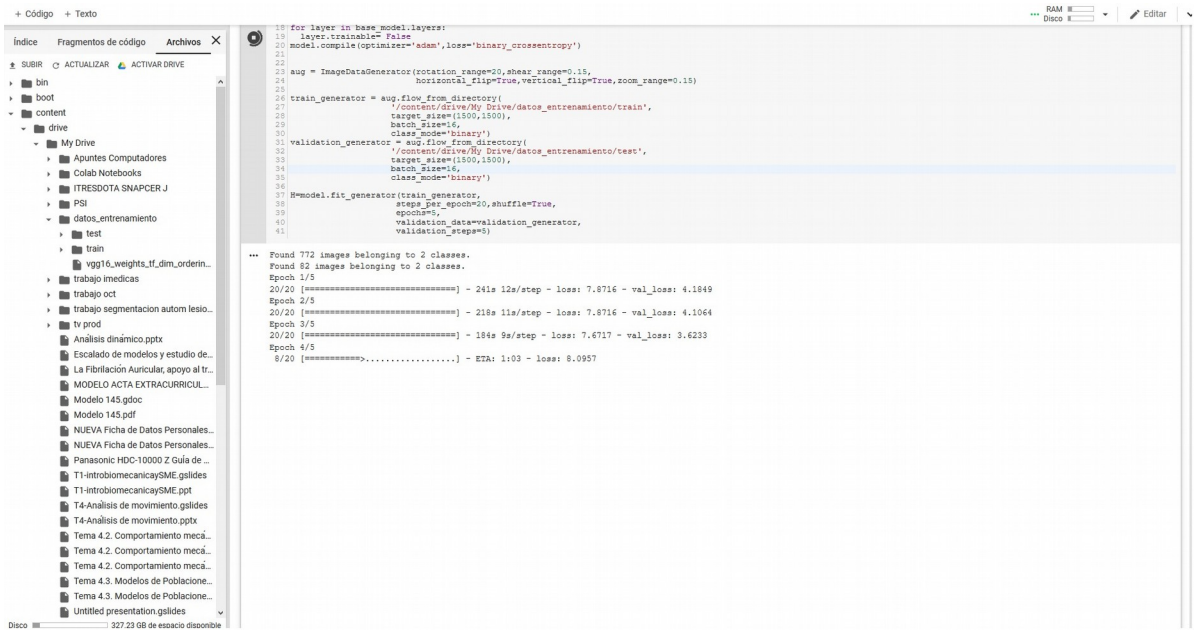


Figura 13: Aspecto de Google Colab

Google Colab asigna al usuario capacidad computacional en la nube de forma opaca a éste en un principio, porque apenas permite seleccionar entre usar la CPU, la GPU o la TPU de la máquina. Sin embargo, lo que el usuario finalmente tiene a disposición es una máquina virtual de una distribución GNU/Linux y un cuaderno de Jupyter, que es el software que la comunidad habitualmente utiliza. Todo esto hace cómodo el paso de un entorno de pruebas local al ofrecido por Google. El servicio pretende hacer cómoda la computación e investigación en el tratamiento de datos al usuario.

Un problema es que Google Colab al ser gratis no garantiza al usuario siempre la misma capacidad de cómputo. Esto provoca que a veces una configuración de entrenamiento resulte viable y que en otros casos se disponga de menor capacidad, todo varía en función de la demanda con lo cual tampoco es sencillo saber en qué momentos va a estar disponible la máquina con las características que son necesarias en un proyecto concreto.

5.2.3 Uso de Spyder

Para poder aprovechar el uso de una máquina especializada, cuestión que se describirá en la siguiente sección resulta bastante útil emplear un programa flexible como es Spyder. Spyder incluye un editor para escribir el código, en el cual están integradas algunas de las herramientas mas habituales de asistencia a la programación, como autocompletado, indentación automática, uso de colores para resaltar estructuras en el código… Además, cuenta el programa con una línea de comandos para poder ejecutar los scripts que deseen usarse así como un navegador de archivos para poder cargar nuevos ficheros con códigos de programación.

Spyder está orientado al uso con Python y en el caso que ocupa en la siguiente sección es recomendado su uso. En contraste con los casos anteriores, ahora se dispone de una máquina potente con lo cual no es necesario trabajar con herramientas orientadas al uso remoto. Así, Spyder explota la idea de “tener todo a la vista” para casi no necesitar utilizar otra aplicación durante el desarrollo de los programas.

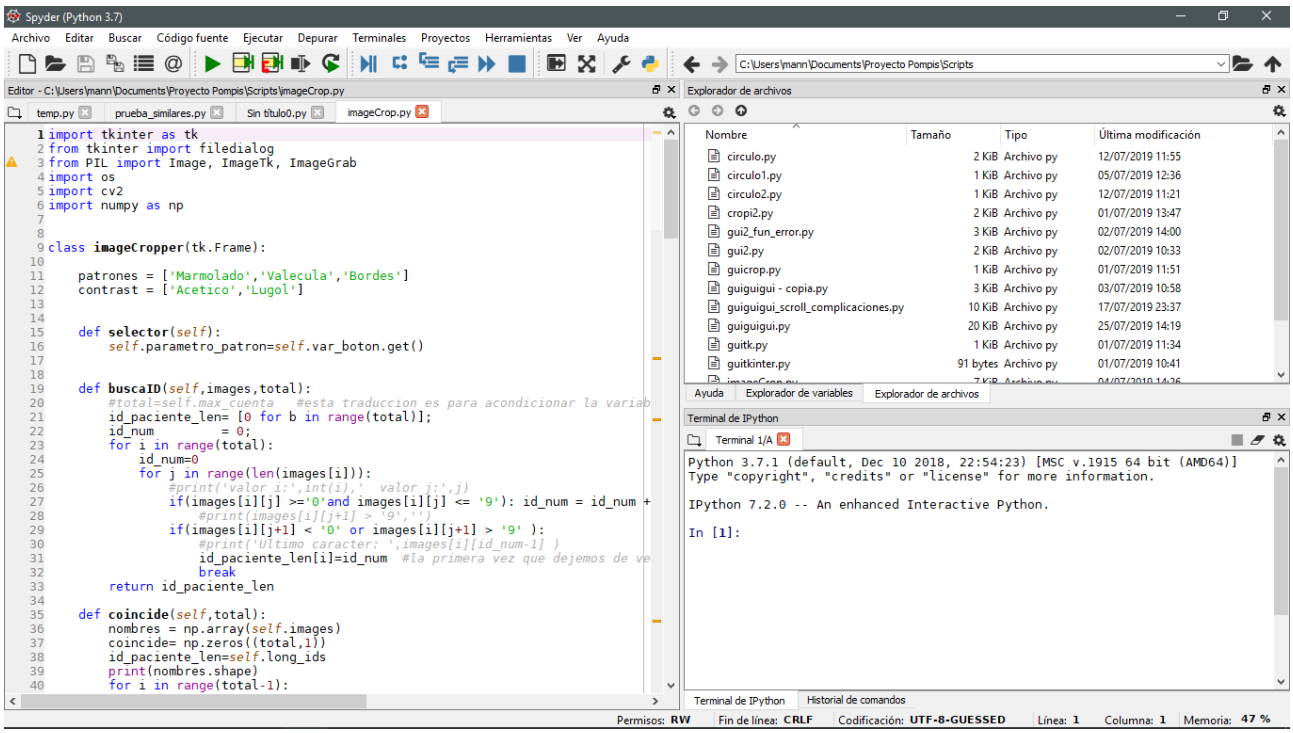


Figura 14: Aspecto del programa Spyder

5.2.4 Entorno local con una máquina especializada

En este apartado se intenta comentar el entorno habitual entre quienes desarrollan técnicas de *deep learning* y AI. Es en el momento en el que se consigue aprovechar los avances en *hardware* y *software* que las técnicas citadas consiguen su mayor popularidad. En la actualidad, de hecho, la mayoría de proyectos relacionados con la visión artificial y el reconocimiento de patrones están trabajando con esta tecnología.

El entorno habitual está fundamentado en la capacidad computacional que ofrecen las GPU, o unidades de procesamiento gráfico, como se ha comentado antes. Y normalmente los equipos de trabajo cuentan con un ordenador de alto rendimiento y potencia que cuenta con una unidad de procesamiento gráfico de alta capacidad. Es así como se ha conseguido popularizar la técnica, porque con esos avances se ha pasado de tiempos de entrenamiento de semanas o meses a días o horas, tal es la diferencia.

La empresa NVIDIA fabrica estos dispositivos y tiene mucha fama entre la comunidad. Tiene modelos con capacidades de entre 6-12 *gigabytes* de memoria gráfica. Tiene a disposición de los usuarios de sus productos cierto software que optimiza el trabajo con los mismos, facilitando la posibilidad de ejecutar pruebas sin tener que realizar tediosas configuraciones de *hardware* y *software*.

Existen alternativas como los dispositivos que produce la empresa AMD, que son de rendimiento comparable a los de las tarjetas NVIDIA o el caso anterior de computación en la nube, con servicios de alquiler de hardware de Google, los servicios de Microsoft Azure, Amazon Web Services...

En general al buscar el dispositivo de tipo GPU ideal deben analizarse ciertos parámetros: los FLOPS acrónimo de *Floating Point Operations Per Second*, una medida de las operaciones que se pueden realizar por segundo, y los núcleos de tensores, que permiten el trabajo especializado en *deep learning* con la posibilidad por ejemplo de trabajar a 16 bits, facilitando el cálculo rápido de combinaciones de matrices. Es habitual también que las tarjetas gráficas dispongan de una gran cantidad de RAM grande, de unos 6Gb en adelante, y con velocidades de reloj cada vez mayores.

Todo esto debe contemplarse para conseguir un entorno de trabajo cómodo, sin dejar de tener en cuenta las características que un ordenador de alto rendimiento también tiene, como por ejemplo suficiente memoria RAM o un procesador eficaz, ya que aunque se delegue en la capacidad de procesamiento gráfico, ésta necesita del

soporte del resto de componentes.

Así, es posible utilizar un computador con características como las comentadas en el Departamento de Teoría de la Señal y Comunicaciones de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla, y tiene concretamente las siguientes especificaciones:

- Procesador intel i9-9700k
- 32 gb de memoria de tipo RAM con una velocidad de 3200 MHz
- Tarjeta Gráfica NVIDIA Titan RTX
- Almacenamiento de 1 TB de memoria

Esta máquina sirve de banco de entrenamiento para las pruebas definitivas del trabajo aquí desarrollado. Es el entorno que mayor comodidad y eficacia ofrece. Se utiliza en dicho ordenador el programa y entorno de Anaconda para que no existan discontinuidades entre el paso de trabajar en un ordenador portátil o trabajar en la nube y trabajar en una máquina de gran potencia.

Existe la posibilidad de utilizar la máquina de forma remota. Esto es enormemente deseable porque los tiempos de entrenamiento en *deep learning* pueden llevar muchas horas o días. Debido a que es un ordenador situado en la Universidad, resulta bastante cómodo poder acceder al mismo en horarios flexibles. Para esto, se trabaja con el programa gratuito TeamViewer. Con esta aplicación es posible acceder a y controlar un ordenador de forma remota con enormes comodidades. El programa despliega una ventana que incluye el escritorio del otro computador en el ordenador desde el cual se está accediendo. Así, se puede aprovechar el tiempo mucho mejor, pudiendo lanzar entrenamientos que duran todo el día, para recabar información por la tarde y ejecutar nuevos entrenamientos durante la noche.

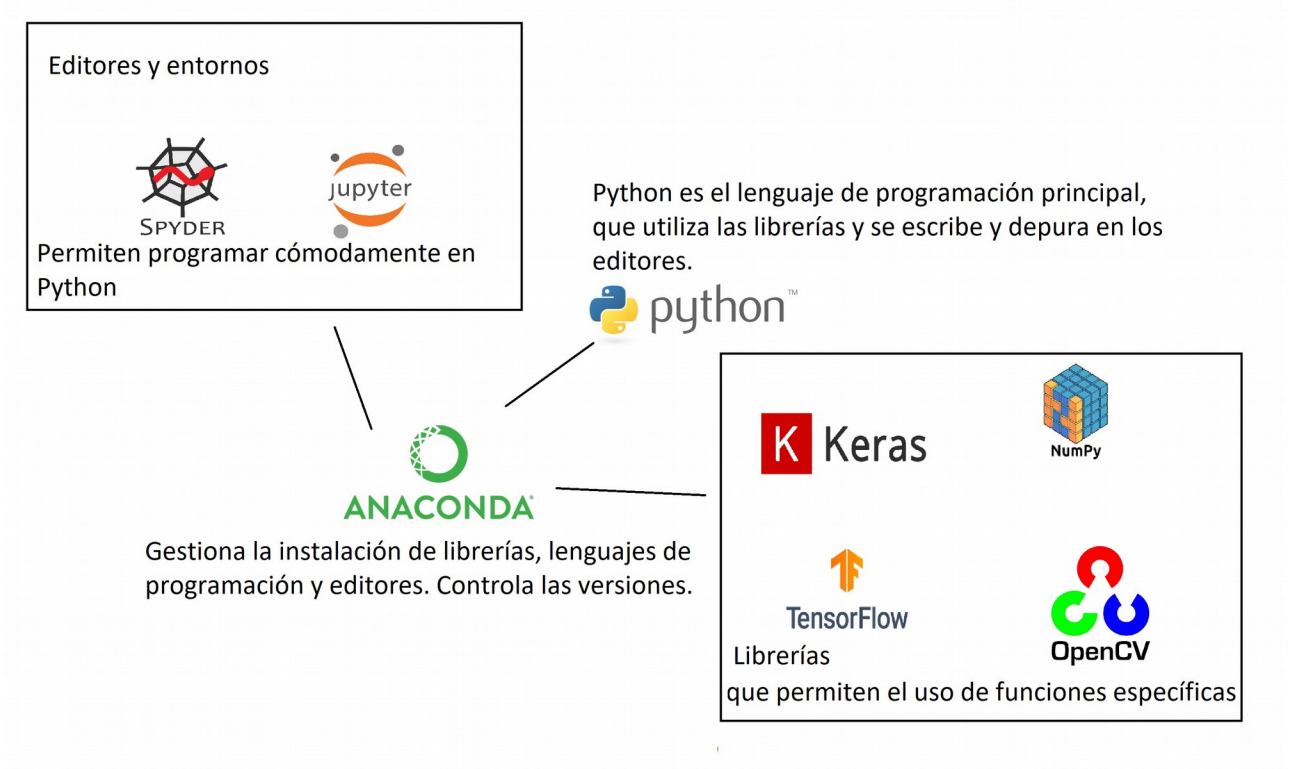


Figura 15: Esquema general de dependencias entre las herramientas usadas

5.3. Organización inicial de la base de datos de imágenes

Al comenzar con el proyecto, tal y como se ha comentado se trata de resolver uno de los primeros problemas, la traducción de los identificadores de las imágenes. Esto se lleva a cabo con varios scripts de Python. Lo primero, como era necesario comparar dos identificadores distintos es ser capaz de extraer dichos identificadores del nombre de las fotos.

El número de historia clínica (NHC) de cada paciente es un identificador diferente para cada centro hospitalario, pero suele tener 6 o 7 dígitos. El número único de historia en Andalucía o NUHSA es común a los centros hospitalarios andaluces, haciendo accesible la información de un paciente desde cualquier hospital. Para evitar tener que calcular repetidamente los identificadores, se hace una primera pasada sobre la lista de imágenes deseadas o contenidas en un directorio y se guarda la longitud que ocupa cada identificador de cada foto, para que cuando sea necesario, se puedan extraer los caracteres conforme a la posición que ocupan en la cadena de texto que es cada nombre de archivo. Esta funcionalidad se desarrolla también para los identificadores NUHSA, que tienen una pequeña diferencia, comienzan con 'AN' y les sigue un número de 10 dígitos. Los cuadros de texto 1 y 2 muestran las funciones que encuentran los identificadores:

```
1.  ##################################################### para detectar longitud de los nuhsa
2.  id_paciente_len= [0 for b in range(total)];
3.  id_num         = 0;
4.  for i in range(total):
5.      id_num=0
6.      for j in range(1,len(images[i])):
7.          if(images[i][j] >='0'and images[i][j] <= '9'):
8.              id_num = id_num + 1;
9.              if(images[i][j+1] < '0' or images[i][j+1] > '9' ):
10.                 id_paciente_len[i]=id_num+2  #la primera vez que dejemos de ver numeros,
termina el id de paciente
11.                 break
```

Texto 1: Código: script para la detección de la longitud de los NUHSA

```
1.  ##################################################  para detectar la longitud de los nhc
2.  id_paciente_len= [0 for b in range(total)];
3.  id_num         = 0;
4.  for i in range(total):
5.      id_num=0
6.      for j in range(len(images[i])):
7.          if(images[i][j] >='0'and images[i][j] <= '9'):
8.              id_num = id_num + 1;
9.              if(images[i][j+1] < '0' or images[i][j+1] > '9' ):
10.                 id_paciente_len[i]=id_num  #la primera vez que dejemos de ver numeros,
termina el id de paciente
11.                 break
```

Texto 2: Código: script para la detección de la longitud de los NHC

Van a pertenecer a un mismo paciente un número de imágenes de entre 3 y 10. Los nombres de archivo tienen indicadores de paciente, fecha y referencias a la existencia o no de zonas de interés o lesión como por ejemplo, un detalle del grado de una displasia. Almacenar toda esta información es necesario para poder organizar y poder seleccionar un sujeto para que se nos desplieguen todas las imágenes que le son relativas. Se intenta entonces agrupar en álbumes las fotografías de cada paciente.

Lo primero entonces, como se ha comentado va a ser comprobar cuántos dígitos del nombre del archivo corresponden a los números de identificación del paciente, para poder luego comprobar en qué archivos tenemos el mismo identificador y agrupar todas las imágenes de un mismo paciente. Se propone generar un fichero de

referencias en el que se almacenen tales cuestiones. Se observa que existen identificadores muy similares, que difieren únicamente en un dígito: e.g. 106578 y 1065781, al tratarse de errores de transcripción, se remite a los profesionales sanitarios para que revisen tales casos.

El conjunto de imágenes tiene archivos con nombres en los que los números iniciales van del 1 al 3, y se separan en carpetas según el primer dígito del identificador. Se mantiene dicha organización ya que cada carpeta contiene un número considerable de imágenes (entre 1000 y 3000) y no hay imágenes de un mismo paciente en distintas carpetas.

El siguiente paso es tratar de encontrar en los nombres de los archivos las fechas de realización de las pruebas. Éstos datos se encuentran siempre en el nombre del archivo justo después del identificador de paciente, con un formato de: 2 o 3 letras iniciales del mes, seguido de cuatro números referidos al año o únicamente los 2 últimos números del año. El script que toma las fechas de los nombres de los archivos se muestra en el siguiente cuadro de texto:

```
1.  nombres = np.array(images)
2.  fecha_paciente_len = np.zeros((total,1))
3.  fechas = [('Fechas \n')]
4.  flag=False
5.  for j in range(total):
6.      for n in range(len(nombres[j])-id_paciente_len[j]-2):
7.          if(nombres[j][id_paciente_len[j]+n].isdigit() and nombres[j][id_paciente_len[j]
+n+1].isdigit() and not nombres[j][id_paciente_len[j]+n+2].isdigit()):
8.              if(nombres[j][id_paciente_len[j]]=='_'):
9.                  fecha_paciente_len[j]=n+2
10.                 fechas.append(str(nombres[j][id_paciente_len[j]+1:id_paciente_len[j]
+int(fecha_paciente_len[j][0])]+' '+str(nombres[j])))
11.                 break
12.             else:
13.                 fecha_paciente_len[j]=n+2
14.                 fechas.append(str(nombres[j][id_paciente_len[j]:id_paciente_len[j]
+int(fecha_paciente_len[j][0])]+' '+str(nombres[j])))
15.                 break
16.         if(nombres[j][id_paciente_len[j]+n].isdigit() and nombres[j][id_paciente_len[j]
+n+1].isdigit() and nombres[j][id_paciente_len[j]+n+2].isdigit()):
17.             if(nombres[j][id_paciente_len[j]]=='_'):
18.                 fecha_paciente_len[j]=n+4
19.                 fechas.append(str(nombres[j][id_paciente_len[j]+1:id_paciente_len[j]
+int(fecha_paciente_len[j][0])]+' '+str(nombres[j])))
20.                 break
21.             else:
22.                 fecha_paciente_len[j]=n+4
23.                 fechas.append(str(nombres[j][id_paciente_len[j]:id_paciente_len[j]
+int(fecha_paciente_len[j][0])]+' '+str(nombres[j])))
24.                 break
25.
26. np.savetxt('fechas.txt',np.array(fechas,dtype=object),fmt='%s')
```

Texto 3: Código: script para la toma de las fechas de realización de las pruebas

Puesto que se consigue filtrar bien la fecha de cada captura, lo siguiente sería tratar de crear conjuntos de imágenes que se han realizado en las mismas fechas y pertenecen al mismo paciente. Debido a que el formato en el que las fechas han sido introducidas tiene consistencia en una misma sesión, será relativamente fácil comparar imágenes de distintas fechas de un mismo paciente, pero imágenes de distintos pacientes en las mismas fechas pueden ser más difícil de agrupar, debido a las inhomogeneidades a la hora de introducir manualmente los datos, que con el transcurso de los años, terminan por referirse a las mismas fechas de forma diferente. Por ejemplo,

podemos encontrar archivos con fechas "Jun15", "jun15", "jn15","jun2015".

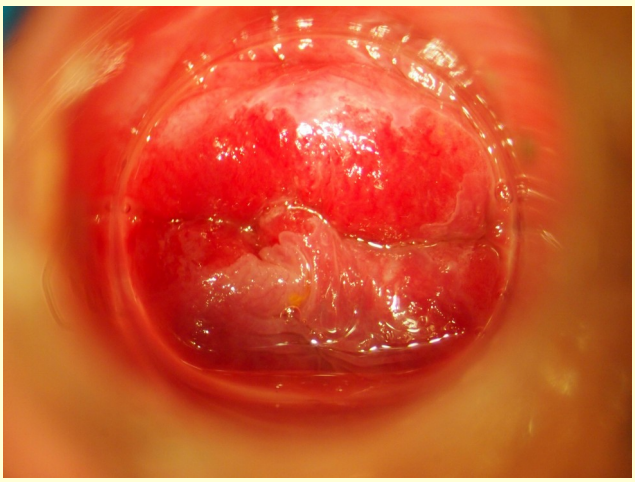
El siguiente paso, necesita agrupar conjuntos de imágenes que incluyan lesiones similares, primero con independencia de fechas y pacientes, pero con la posibilidad posterior de filtrar precisamente por fechas y pacientes. Por ejemplo: seleccionar todas las displasias de grado 2, se retienen únicamente las imágenes a las que incluyan "AIN2"; luego, podría ser deseable seleccionar únicamente las que se centran en un paciente concreto, 123456 y de ese paciente podrían buscarse las capturas realizadas en una fecha única; otra utilidad puede ser presentar imágenes de otros pacientes que tienen una misma referencia para la lesión. Existen muchas otras posibles utilidades que pueden ser deseables por el personal médico, por ejemplo: tratar de comparar imágenes que tengan las mismas lesiones, y que sean de distintos pacientes pero que éstos se encuentren bajo similares procesos de evolución de la enfermedad, por ejemplo buscar otros pacientes que han pasado de un grado inferior al que ahora se estudia en periodos de tiempo similares.

Tal y como se ha comentado, la siguiente función a realizar es una que extraiga de las cadenas de texto los códigos “AIN1”, “AIN2”, “AIN3”. Como en este punto se ha guardado la longitud del identificador de paciente y se conoce cuánto ocupa la fecha, la extracción de las observaciones acerca del grado de la lesión deben encontrarse justo después de los otros campos, esto facilita la búsqueda y permite reutilizar el código anterior.

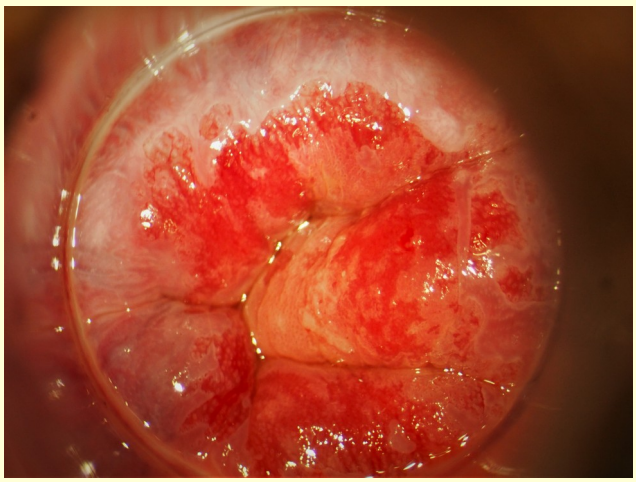
Tras estos pasos, se pasa a desarrollar una función que dado un NHC retorne el NUHSA correspondiente. Tras obtener un listado de tales equivalencias en el que se tienen los identificadores de todos los pacientes relacionados con el proyecto POMPIS, podrá automatizarse la traducción de los mismos. La función que la realiza genera una lista de todos los nombres de archivo existentes en un directorio. Calcula la longitud que ocupa el identificador, lo selecciona y busca una coincidencia en otra lista, la que tiene las equivalencias entre NHC y NUHSA. Seguidamente se guarda una copia de dicho archivo, sustituyendo el nombre por: el identificador equivalente de la lista y después el resto de información de la foto original. Por ejemplo, si teníamos el archivo “1234567jun15_AIN1_a.jpg”, primero descubrimos que el NUHSA equivalente es “AN0012345678”, así que generamos una imagen igual con nombre “AN0012345678jun15_AIN1_a.jpg” y así sobre todas las imágenes que existan en el directorio. Las imágenes que no pueden crearse, por encontrarse corrompidas, o por no existir un identificador equivalente generan una excepción y se incluyen en una lista de texto, para que los profesionales puedan revisarlas. Esta función tiene el aspecto mostrado en el texto 4.

```
1.  equivalencias     = np.genfromtxt('NUHSA_NHC.csv',skip_header=1,delimiter=',',dtype='str')
2.  path_imagen_nueva='CambiaID'
3.  path_imagen_nueva=os.path.join(path,path_imagen_nueva)
4.  NHCsinNUHSA = []
5.  print(type(equivalencias[0][1]))
6.  for i in tqdm(range(total)):
7.      dato=nombres[i][0:id_paciente_len[i]]
8.      imagen=cv2.imread(os.path.join(data_path,images[i]),1)
9.      coincidencia = np.where(np.any(equivalencias == nombres[i][0:id_paciente_len[i]] ,
axis=1))
10.     if(coincidencia[0].size > 0):
11.         if(coincidencia[0][0] >0):
12.             print(i)
13.             if not
cv2.imwrite(os.path.join(path_imagen_nueva,str(equivalencias[coincidencia[0][0]][0])+
14.                                     str(nombres[i][int(id_paciente_len[i]):])),imagen):
15.                 raise Exception("No pudo crearse la imagen")
16.         else: print('Paciente no encontrado: '+str(nombres[i][0:id_paciente_len[i]]))
17.     else:
18.         NHCsinNUHSA.append(str(dato))
19. np.savetxt('NHCsinNUHSA3.txt',NHCsinNUHSA,fmt='%s')
```

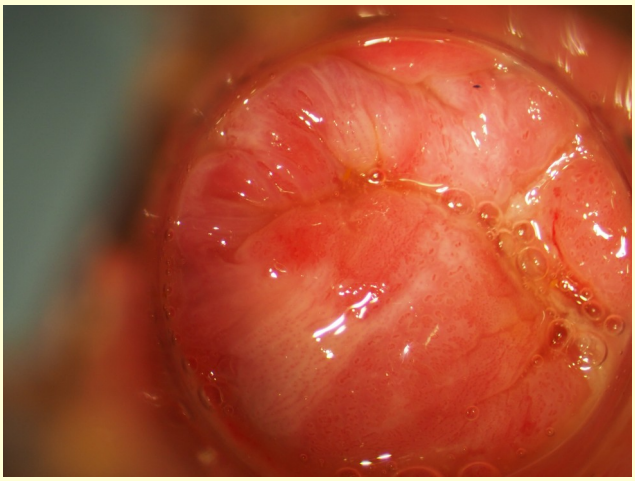
Texto 4: Código: script para generar las imágenes de la base de datos con los identificadores cambiados de NHC a NUHSA.



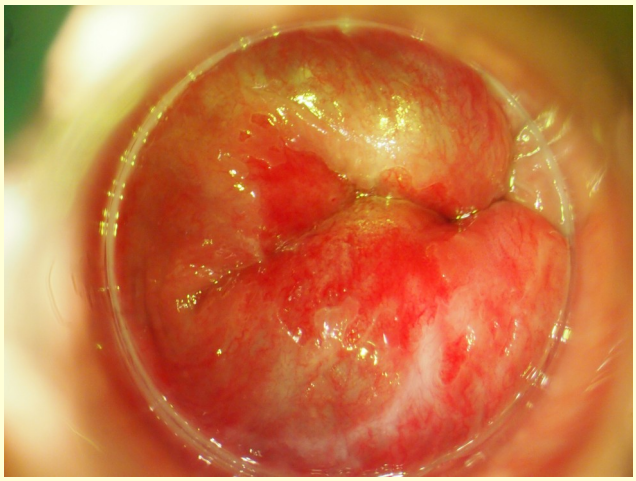
a)



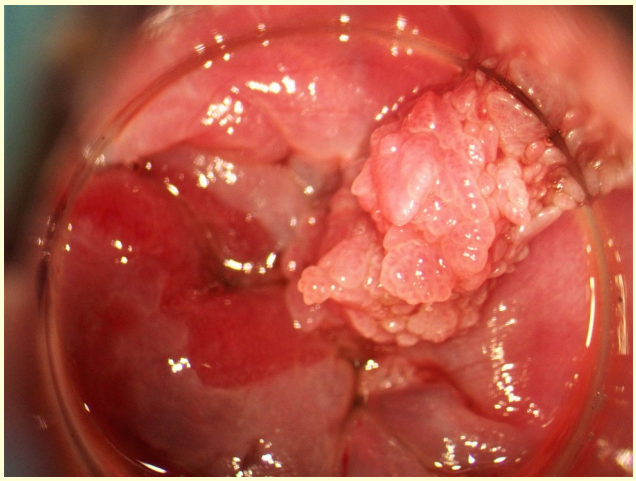
b)



c)



d)



e)

Tabla 4: Aspecto de imágenes con resultados de biopsia distintos. De izquierda a derecha y de arriba abajo: a) imagen normal, b) imagen de displasia de grado alto, c) imagen de displasia de grado medio, d) imagen de displasia de grado bajo, e) imagen de condiloma

Tras realizar la traducción de imágenes, se notifica que efectivamente, hay identificadores que no se encuentran, y la base de datos queda reducida en unas 1200 imágenes, a la espera de encontrar los ids que faltan. Esto como se ha mencionado, puede deberse a un error de transcripción durante la fase de captura y almacenamiento de las imágenes, ya que es un proceso que los médicos han realizado a mano. Por suerte, como toda la traducción está ya bastante automatizada, con incluir las equivalencias entre identificadores, se recuperarán dichas imágenes sin dificultad.

En un análisis inicial, no se detecta un criterio para filtrar imágenes, en cuanto a si resultan de utilidad o no para extraer patrones, así que no pueden retirarse imágenes inicialmente, ya que debe observarse una a una para poder decidir si cada imagen es buena o mala candidata para extraer un recorte.

5.4. Desarrollo de la aplicación de recortes de imágenes ImageCropper

Llegado a este punto, el proyecto presenta una base de datos con la gestión suficiente para comenzar a trabajar en la extracción de recortes de patrones relevantes. Existe la posibilidad de seleccionar imágenes relativas a un paciente, se tienen sus fechas de intervención y el grado de su enfermedad. Desgraciadamente, la técnica de la Anoscopia de Alta Resolución es difícil de dominar, y la pequeña referencia que los médicos dejan en el nombre de archivo de las imágenes no es suficiente información para generar recortes a modo de verdades de referencia, por lo tanto se necesitan más datos. La base de datos de imágenes se acompaña entonces de las Hojas de Evolución, en las que los médicos describen en texto (digitalmente o en papel) la intervención clínica. Lo que más relevante resulta extraer de las hojas de evolución para el desarrollo del proyecto es la toma de biopsias. Se establece que el resultado de una biopsia es la verdad de referencia, así que también se necesitan datos de los resultados de las mismas. Una vez se tiene la colección de Hojas de Evolución y los resultados patológicos, se deberá extraer de los mismos los identificadores para que dada una imagen, sea posible leer la descripción de la intervención clínica y comprobar los resultados de la misma. Un problema que surge ahora es que no de todas las intervenciones se tienen datos de los tres tipos, muchas imágenes son de intervenciones que no tienen hoja de evolución, con lo cual sin descripción clínica no se sabe de que zona de la imagen extraer el recorte, o sin resultado de la biopsia tampoco se puede tomar por cierto sin dudar el diagnóstico del médico. Estos casos en los que faltan datos se anotan manualmente para solicitar a los profesionales la incorporación de las cuestiones que falten.

Para tratar de realizar la extracción de parámetros con celeridad, se plantea y realiza una aplicación con interfaz gráfica en la que no sea necesario calcular las coordenadas de la imagen en la que se va a realizar el recorte.

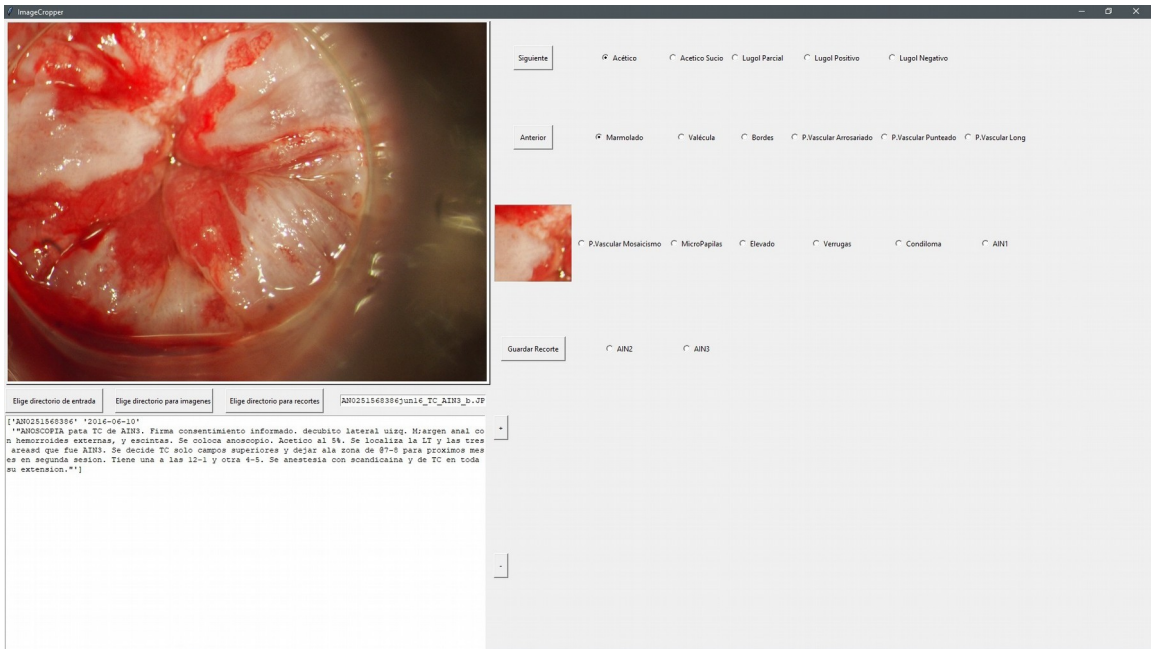


Figura 16 Aspecto de la aplicación ImageCropper en un estado inicial.

Se ha comentado que está a disposición la lista de las hojas de evolución, en las que existe una descripción de la intervención que el médico realiza a un paciente. La idea es identificar las zonas en las que el médico detecta la lesión y utilizar dicho desarrollo para encontrar la región de interés y hacer el recorte con un click del ratón sobre la imagen. Parte de la motivación para realizar esta aplicación es que fuera utilizada por los médicos, ya que ellos con mucha seguridad podrán realizar la labor de extracción de recortes, con la ayuda extra de que la aplicación tiene un uso muy sencillo: dada una imagen, ésta se presenta en el visor principal, que en la figura anterior se encuentra a la izquierda, el usuario sigue la descripción del médico que existe en el cuadro de texto situado bajo el

visor principal, activa los selectores de los patrones que el profesional describa, se localiza la zona indicada por el médico, se hace click con el botón izquierdo del ratón y se visualiza en el visor secundario, situado a la derecha del visor principal en la figura anterior. Finalmente y si el recorte se da por bueno, se pulsa en guardar recorte para conseguir almacenarlo en un directorio.

La figura anterior, es una captura de un estado inicial de la aplicación, a modo de prototipo, debido a que era necesario realizar un poco de experimentación e investigación ya que un servidor jamás ha realizado aplicaciones o programas con GUI (*Graphical User Interface*). Estando todo el proyecto bajo la bandera de Python, la aplicación ImageCropper no iba a ser diferente. Tras realizar una búsqueda y comparativa entre las librerías que Python tiene para programar interfaces gráficas, se decide usar **tkinter**. Existen alternativas como wxPython y PyQt, pero la elección de Tkinter se realiza con un criterio relajado: es tan utilizada que se ha convertido en un estándar, tiene una documentación algo escasa pero eso es habitual en librerías gráficas y sin embargo, la comunidad tiene a disposición de los usuarios miles de ejemplos de mayor o menor calado y complejidad, así como tutoriales para comenzar desde cero, además, Tkinter es la biblioteca que viene por defecto en la versión de Python para Microsoft Windows.

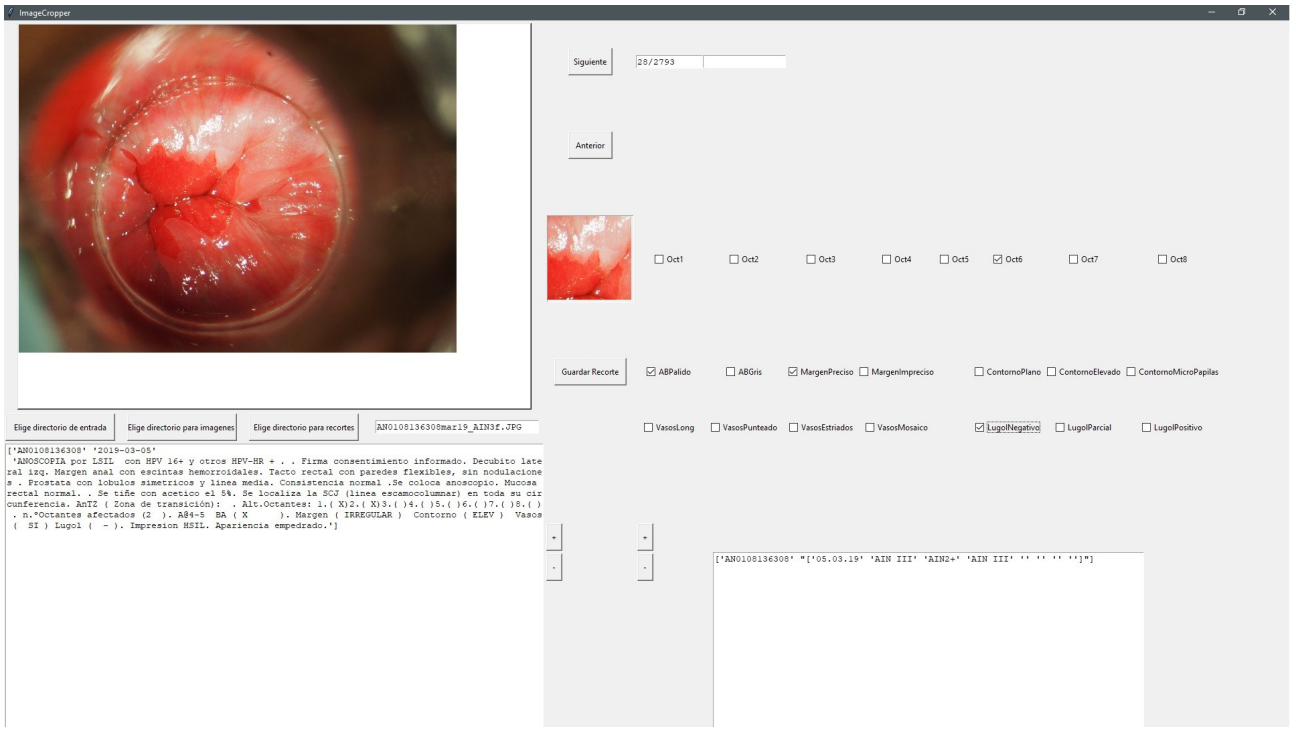


Figura 17: Aspecto de la aplicación ImageCropper en un estado más avanzado de desarrollo.

Como puede observarse en la figura 17 sobre este párrafo, la aplicación termina por incluir también un cuadro de texto para la información de los resultados de las biopsias, cuestión esencial que en el prototipo se obvió para poder centrar el esfuerzo en la parte relativa al manejo de las imágenes. También se actualiza el método de selección de patrones, para asemejarlo a un formato que los médicos tratan de instaurar, y que queda reflejado en la tabla de la figura siguiente. Vemos que existe una división en zonas, que se corresponden con las filas de la tabla. Habrá filas según la división del anoscopio que los médicos deseen usar. La subdivisión más común es la de octantes. Se divide el área de inspección del anoscopio en octantes para poder contrastar con otros profesionales zonas concretas afectadas por la enfermedad. En la tabla propuesta, los médicos habrán de marcar aquellas casillas en las que consideren que existe el patrón especificado. Si las filas se correspondían con la localización, las columnas tendrán la información de la lesión y los patrones visuales que la describen. De izquierda a derecha, la tabla plantea la identificación de los siguientes patrones: AB, de Blanqueamiento por Ácido Acético, podrá ser P, de pálido o G, de Gris; el Margen de la lesión podrá ser, P, de preciso, o I de impreciso; el Contorno, Pl, de plano, @ de elevado o MP de micropapilas; los Vasos Sanguíneos serán L, de longitudinal, P, de punteado, E, de estriado o M, de marmolado; la respuesta al Lugol será, N de negativa, ± de parcial, o P, de positiva. Además, se indicará qué biopsia se realizó y donde, para saber que el resultado a la primera biopsia se corresponde con una muestra en la posición indicada en la primera biopsia, para lo cual se utiliza otra subdivisión diferente del círculo

del anoscopio: la división en doce, inspirada en la referencia de un reloj analógico. Un ejemplo de una referencia a una biopsia sería C@4, lo cual significa que la biopsia C, es decir, la tercera según el orden alfabético se ha realizado a las 4 horas.

	AB		Margen		Contorno			Vasos				LUGOL			N Biop	Impres	Resul	NOTAS
	P	G	P	I	PI	@	MP	L	P	E	M	N	+/-	P	@	Clinica	Biopsia	
Left Lateral																		
Left Anterior																		
Anterior																		
Right Anterior																		
Right Lateral																		
Rigth Posterior																		
Posterior																		
Left Posterior																		

Figura 18: Tabla propuesta por el personal del HUVR para agilizar el diagnóstico y la comparación de las anoscopias.

El desarrollo del programa conlleva crear diversas funciones para las tareas. Primero, tkinter tiene muchas opciones para gestionar el espacio de la interfaz. Para los visores, se emplean lienzos, llamados *canvas* en la librería, para poder controlar opciones de presentación. Sobre los lienzos se imprimen las imágenes, la original y el recorte. Los cuadros de texto son 5 en total. Bajo el visor principal está el de mayor tamaño, que contiene la información de las hojas de evolución. El segundo visor es para los resultados de biopsias. Estos dos primeros visores cuentan con botones para cambiar sus contenidos, ya que de un mismo paciente pueden existir varias sesiones, de forma que se puede ver el desarrollo. El tercer cuadro de texto muestra el nombre del archivo de la fotografía que se expone en el visor principal y se encuentra también debajo del mismo. El cuarto cuadro de texto muestra el número de imagen actual y el total del directorio actual. El último cuadro de texto es una entrada, de forma que introduciendo un valor menor que el total de imágenes que se muestra en el cuarto cuadro de texto el visor mostrará la imagen almacenada en dicha posición.

Existen seis botones mas. Hay tres bajo el visor principal para seleccionar los directorios. El situado más a la izquierda es para elegir las imágenes de anoscopia, el central es para guardar una copia de la imagen original, con la idea de generar un directorio de imágenes usadas en recortes, y el botón situado a la derecha de los anteriores es para elegir el directorio en el que guardar los recortes. Bajo el visor secundario hay otro botón que se utiliza para guardar el recorte que se muestra. Los dos últimos botones son para cambiar la imagen del visor principal a la siguiente o la anterior. La botonera de selectores de la derecha de la interfaz es como se ha mencionado para darle el nombre a los recortes, la primera fila tiene los octantes y las demás los patrones.

Tkinter permite controlar la posición de los elementos que se muestran con varias herramientas. En la aplicación se utiliza *grid*. Con *grid* se puede posicionar todo como en una rejilla. La interfaz queda entonces dividida en cuadros, formando una tabla donde pueden colocarse los elementos. Cuando un objeto es muy grande, debe tenerse en cuenta que si no se especifica, ocupará siempre una casilla de la tabla, dejando la fila y la columna deformadas para ocupar el alto y el ancho del objeto. Así que se debe indicar que un objeto es de gran tamaño indicando que sobresale varias filas o columnas de forma que se puedan colocar elementos de menor tamaño junto al grande aprovechando correctamente el espacio. Para visualizar esto, en las capturas del aspecto de la interfaz, puede verse como bajo el visor principal hay cuatro botones y un cuadro de texto. Esos elementos son de igual anchura que el visor, así que el visor ocupa cuatro casillas de la rejilla, con lo cual el cuadro de texto de las hojas de evolución también. Si dichos elementos no se hubieran configurado para ocupar varias casillas, entre ellos solo podría introducirse otro más.

En cuanto a la generación del recorte, el uso de **OpenCV** es crucial. Las funciones necesarias para la muestra de las imágenes pertenecen a la mencionada librería. Con el uso del *canvas* (lienzo) de tkinter, se puede capturar la posición del cursor dentro del mismo y así generar un evento cuando se haga click izquierdo con el ratón sobre el visor principal, tomando un recorte que se mostrará en el visor secundario. Para empezar, las funciones de cambio de imagen, siguienteImagen y previaImagen, controladas por los botones de “Siguiente” y “Anterior”, son las que agregan el **evento** que se disparará al hacer click con el ratón sobre la imagen y lo hacen para cada nueva imagen. Cuando se accede a la función que gobierna el evento, se captura la posición del ratón y se llama a otra función, **cropCrea**. Dicha función trata de crear el recorte sin que éste se escape de los bordes de la imagen, de forma que si el usuario realiza un corte justo al borde, el centro del recorte se situará desplazado hacia el centro de la imagen. Concretamente verifica si la posición vertical y horizontal son mayores que cero al restarles la mitad del tamaño del recorte, ya que el recorte está centrado en la posición del ratón. En caso de realizar un recorte muy al borde, la

posición por ejemplo horizontal menos la mitad del tamaño del recorte sería inferior a cero, es decir, el recorte se sale de la imagen. En lugar de aceptar dicha posición y rellenar con ceros ese espacio del recorte, se decide por truncar la posición del recorte a la más al borde posible, es decir, la mitad del tamaño de recorte. Se muestra ahora la definición de la función que gestiona lo que se comenta.

En la primera línea, vemos que la función toma como parámetros las posiciones horizontal y vertical que dispararon el evento, tam es una variable que almacena el tamaño del recorte que va a realizarse, que será cuadrado, img almacena la imagen de la que se desea realizar el recorte, la cual estará visible en el visor principal y finalmente self hace referencia a qué función llama a la definición de cropCrea, de forma que se pueden acceder así a variables que no son globales para toda la aplicación si se puede acceder al estado de las variables desde un nivel de abstracción superior. Los condicionales de las líneas 3 y 9 comprueban que la posición del cursor no está situada demasiado a la izquierda o demasiado arriba de la imagen. Los de las líneas 6 y 12 controlan que la

```
20. def cropCrea(self,posx,posy,tam,img):     #acepta 3 ints de posicion del recorte y tamaño y
                                              #una imagen leida por opencv
21.     hei,wid,chan=img.shape
22.     if(posx-tam/2 <= 0):
23.         posx = tam/2
24.         print('izq')
25.     if(posx+tam/2 >=wid):
26.         posx=wid-tam/2-1
27.         print('der')
28.     if(posy-tam/2 <=0):
29.         posy=tam/2
30.         print('arr')
31.     if(posy+tam/2 >=hei):
32.         posy=hei-tam/2-1
33.         print('aba')
34.     print("Coordenadas:posx,posy",posx,posy)
35.     self.posx = posx
36.     self.posy = posy
37.     self.tam  = tam
38.     self.hei  = hei
39.     self.wid  = wid
40.     self.chan = chan
41.     print("Tam imagen :",str([posy-tam/2,posy+tam/2,posx-tam/2,posx+tam/2]))
42.     return img[round(posy-tam/2):round(posy+tam/2),round(posx-tam/2):round(posx+tam/2)]
```

Texto 5: Código de la función cropCrea

posición no sea demasiado a la derecha horizontalmente o demasiado abajo verticalmente. De la línea 16 a la 21 se actualizan los valores calculados en el entorno desde el que se llamó a la función, de forma que las modificaciones son accesibles fuera de la función. Finalmente, se devuelve por la salida estándar de la función el recorte realizado sobre la imagen, seleccionando aquella región de la variable img que tiene un cuadrado entorno a la posición dada por (posy,posx).

El resultado de la función cropCrea es el recorte, que es usado por guardaCrop al activarse pulsando el botón de "Guardar Recorte". Con el uso de OpenCV puede guardarse una imagen de formato JPG de forma sencilla. Se tiene una estructura de directorios para almacenar los recortes. Existe un directorio raíz llamado recortes, en el cual se sitúan los directorios hijos: AIN3,AIN2,AIN1,Normal,Condiloma. Como se ha comentado, al guardar el recorte se genera un nombre que comienza por el identificador del paciente, sigue con los octantes de los que se recortó y un indice numérico para distinguir recortes diferentes tomados de la misma imagen. Además, al guardar una muestra se genera una entrada en un fichero de texto con las coordenadas del recorte. Así, pueden realizarse recortes de mayor tamaño o de diferentes formas que el que se guarda por defecto. En la actualidad es habitual entrenar redes de clasificación con patrones de diferentes tamaños para comprobar cuál caracteriza mejor la clase. Si el tamaño del recorte es muy grande incluirá el fondo y otros elementos colindantes que dificultarán la identificación, y si es demasiado pequeño podría no captar todos los detalles relevantes con lo cual es provechoso

probar con varios tamaños. El tamaño por defecto que recorta la aplicación es de 128 por 128 píxeles. Debe tenerse en cuenta que para la visualización y manipulación las imágenes tienen el tamaño reducido.

Entonces, las coordenadas guardadas incluyen los siguientes campos: posición horizontal, posición vertical, tamaño del cuadrado de recorte,altura de la imagen redimensionada, anchura de la imagen redimensionada, número de canales de la imagen original, el nombre de la imagen original, los octantes afectados en el recorte y los patrones señalados por el profesional. Con estos datos se realiza otro script para realizar los mismos recortes con imágenes de cualquier tamaño.

Los botones selectores de patrón y octante se encuentran referenciados a una variable que asume un valor al ser activados. Cuando se hace click sobre uno, la variable correspondiente tomará un valor que únicamente se obtiene al activar un selector, de forma que las variables tomarán valores enteros creciendo una unidad cada selector. Por otra parte, se tienen dos vectores, o listas, según el tipo de datos de Python, que tienen almacenados en cadenas de texto los nombres de los patrones. De esta forma, los patrones pueden ser catalogados con cualquier combinación de nombres de entre los disponibles. Debe mencionarse que algunas de esas combinaciones son incompatibles en la realidad, pero el usuario debe seguir las indicaciones del profesional, que en la hoja de evolución ha de dejar claro la guía y la descripción de la región a recortar. Entonces cada vez que un selector es pulsado, se hace una llamada a una función que en un bucle recorre todas las variables de los botones selectores, que se almacenan en una lista para poder acceder a ellas con velocidad . Dentro del bucle, identifica cuales se han activado y han pasado a tomar su valor único, así, las variables que no se activen tienen un valor de cero y el resto, sumará a la variable cadenaNombrePatrones el nombre del patrón que se almacenaba en el correspondiente vector en el índice con el valor de la variable activada. Un ejemplo: se pulsa el selector cuatro, MargenImpreciso, entonces la variable de activación var4 pasará de valer 0 a 4. Seguidamente, en la función que gestiona los selectores, se recorre la lista de variables y se suma al valor de la variable cadenaNombrePatrones el valor del índice 4 del vector de nombres de patrones.

Con el uso de la aplicación, el usuario puede ir trabajando con las fotografías, las hojas de evolución y los resultados de las biopsias a la vez. Puede ir pasando de una en una las imágenes y al cambiar de paciente, se actualizan los cuadros de texto de las hojas y las biopsias, con lo cual resulta cómodo leer los resultados y la descripción del médico para encontrar correctamente la posición en la que se tomaron las biopsias y realizar el recorte. Además, siguiendo la descripción del médico, el usuario puede seleccionar los patrones visuales de los tejidos así como el octante al que pertenece el recorte para que se agreguen al nombre del archivo en el que se almacenará toda esa información. De manera que automáticamente se genera el nombre del recorte al guardarlo, teniendo una estructura así: primero está el identificador tipo NUHSA del paciente, después el o los octantes de interés, seguidos de los patrones de la tabla que el médico haya indicado y finalmente un índice numérico que varía para distinguir capturas de nombres iguales. Se guardarán según el resultado de la biopsia en un directorio distinto: resultado AIN3, AIN2, AIN1, Normal y Condiloma.

Esta es la base del sistema de extracción de patrones. En el estado desarrollado hasta aquí ya existe una herramienta que puede servir para ayudar a aprender a médicos que se encuentren formándose en el análisis de HRA. Con la vasta cantidad de imágenes y resultados de biopsias disponibles así como la guía del profesional que realizó el análisis, un usuario inexperto puede nutrirse de muchos ejemplos, comprobar la evolución de casos concretos a lo largo del tiempo y sobre todo asimilar parte de la experiencia de los exámenes clínicos disponibles.

5.5. Extracción de recortes automatizada con el uso de las coordenadas

Como se ha mencionado, durante la extracción de recortes se genera un fichero en el que se incluyen datos para volver a realizar un recorte en la posición dictada por los profesionales pero modificando ciertos aspectos. Tener guardada la posición de extracción es enormemente útil. Las imágenes de las que se extrae el recorte son redimensionadas para poder trabajar cómodamente con ellas. A la hora de visualizarlas en la aplicación ImageCropper resulta conveniente ya que el tamaño original de las fotografías llega hasta los cuatro mil píxeles de ancho. Esto conlleva usar mucha memoria para tratar con muchas imágenes, razón por la cual se decide a cambiar su tamaño. Realizar los recortes únicamente sobre las imágenes de pequeño tamaño sería infrautilizar la gran calidad de las originales, lo cual motiva la extracción posterior basada en las coordenadas.

Desgraciadamente, durante el diseño de la aplicación se incurrió en algunas deficiencias. Inicialmente no se incluyó una funcionalidad que a las coordenadas agregase el nombre del archivo de la foto original de la que el recorte fue extraído. Ese conjunto de coordenadas son ya inservibles porque habría que buscar recorte a recorte la imagen original a la que pertenecen, cosa que resulta excesivamente tediosa si además se tiene en cuenta que son muy pocos los recortes cuyas coordenadas se encuentran en esta situación. La otra deficiencia es que no se incluye en la mayoría de las coordenadas guardadas el resultado de la biopsia. Esto es debido a que los recortes son guardados en carpetas diferentes según el resultado de la biopsia, lo cual en un principio hace innecesario agregar el resultado, pero a la hora de generar los nuevos recortes resulta mucho mas cómodo. Es por esto que se realiza un script de Python que añade al fichero de coordenadas el resultado de la biopsia de cada recorte. Para esto crea listas de los nombres de las imágenes que están en las carpetas de recortes y busca las correspondencias entre el fichero de coordenadas y los recortes, para que en función de la carpeta en la que están les asigne un resultado de biopsia u otro. Todo está desarrollado en el bloque de código del texto 6. Así se consigue pasar del aspecto del fichero de coordenadas mostrado en la figura 19 al de la figura 20.

```
[[215, 192, 128, 432, 576, 3, 'AN0447868195Feb14_f.JPG', 'Oct6_Oct7_', 'ABPalido_LugolParcial_']]
[[327, 367.0, 128, 432, 576, 3, 'AN0448231038jun16_AIN1_b.JPG', 'Oct6_Oct7_', 'ABPalido_LugolNegativo_']]
[[173, 193, 128, 432, 576, 3, 'AN0464930293Abr14_N_a.jpg', 'Oct7_', 'ABPalido_MargenImpreciso_ContornoPlano_VasosEstriados_LugolPositivo_']]
[[180, 95, 128, 432, 576, 3, 'AN0464930293Abr14_N_b.jpg', 'Oct7_', 'ABPalido_MargenImpreciso_ContornoPlano_VasosEstriados_LugolPositivo_']]
[[286, 231, 128, 432, 576, 3, 'AN0464930293jun18_N_e.JPG', 'Oct7_', 'ABPalido_MargenImpreciso_ContornoPlano_LugolPositivo_']]
[[279, 136, 128, 432, 576, 3, 'AN0464930293jun18_N_a.JPG', 'Oct7_', 'ABPalido_MargenImpreciso_ContornoPlano_LugolPositivo_']]
[[318, 180, 128, 432, 576, 3, 'AN0464930293jun18_N_f.JPG', 'Oct7_', 'ABPalido_MargenImpreciso_ContornoPlano_LugolPositivo_']]
```

Figura 19: Detalle del fichero de coordenadas sin incluir las biopsias

```
[[215, 192, 128, 432, 576, 3, norm 'AN0447868195Feb14_f.JPG', 'Oct6_Oct7_', 'ABPalido_LugolParcial_']]
[[173, 193, 128, 432, 576, 3, norm 'AN0464930293Abr14_N_a.jpg', 'Oct7_', 'ABPalido_MargenImpreciso_ContornoPlano_VasosEstriados_LugolPositivo_']]
[[180, 95, 128, 432, 576, 3, norm 'AN0464930293Abr14_N_b.jpg', 'Oct7_', 'ABPalido_MargenImpreciso_ContornoPlano_VasosEstriados_LugolPositivo_']]
[[286, 231, 128, 432, 576, 3, norm 'AN0464930293jun18_N_e.JPG', 'Oct7_', 'ABPalido_MargenImpreciso_ContornoPlano_LugolPositivo_']]
[[279, 136, 128, 432, 576, 3, norm 'AN0464930293jun18_N_a.JPG', 'Oct7_', 'ABPalido_MargenImpreciso_ContornoPlano_LugolPositivo_']]
[[318, 180, 128, 432, 576, 3, norm 'AN0464930293jun18_N_f.JPG', 'Oct7_', 'ABPalido_MargenImpreciso_ContornoPlano_LugolPositivo_']]
```

Figura 20: Detalle del fichero de coordenadas con las biopsias incluidas

Puede verse que entre ambas figuras hay una línea de coordenadas que ha sido omitida. El script guarda en otro fichero distinto los recortes a los que no ha sido capaz de agregar información de la biopsia. Esto es resultado de un filtrado de recortes, que durante la inspección posterior a su obtención son eliminados, debido principalmente a una mala visibilidad o un mal etiquetado.

En el texto 6 se pueden ver algunas útiles posibilidades de programación que Python ofrece al usuario. Por ejemplo en las líneas de la 27 a la 30 se seleccionan el identificador, los octantes y los patrones del recorte de una forma similar a la que permitiría M. Gracias al uso de la librería numpy, el manejo de índices es muy cómodo, ofreciendo posibilidades como en las líneas 28 y 29 en las que se pueden tomar los valores que pertenecen a un campo y ocupan desde un punto inicial hasta los dos o tres últimos valores de ese campo. En el caso de campos de longitud variable, como el caso de octantes y patrones, se puede capturar información de forma precisa y rápida.

La razón por la cual no se selecciona el campo completo es que, como puede comprobarse en las figuras 19 y 20, al guardar las coordenadas se incluyen comas para separar los campos y comillas para delimitar aquellos que contienen texto, así que con el paso anteriormente descrito se evita tener que eliminar o delimitar con mayor esfuerzo estos elementos.

```
coordenadas = np.genfromtxt('coordenadas1212.txt',dtype='str')
data_path = 'HPV_Imagen'
reco_path = 'recortes'
path_ain1  = 'Biopsias con resultado AIN1'
path_ain2  = 'Biopsias con resultado AIN2'
path_ain3  = 'Biopsias con resultado AIN3'
path_norm  = 'Biopsias con resultado Normal'
path_cond  = 'Biopsias con resultado Condiloma'
path_norB  = 'Biopsias Normales sin h'
data_path = os.path.join(data_path,reco_path)
listaain1 = os.listdir(os.path.join(data_path,path_ain1))
listaain2 = os.listdir(os.path.join(data_path,path_ain2))
listaain3 = os.listdir(os.path.join(data_path,path_ain3))
listanorm = os.listdir(os.path.join(data_path,path_norm))
listacond = os.listdir(os.path.join(data_path,path_cond))
listanorB = os.listdir(os.path.join(data_path,path_norB))
listotal  = listaain1 + listaain2 + listaain3 + listanorm + listacond + listanorB
total     = len(listotal)
total2    = len(coordenadas)
coorde    = []
coorde2   = []
coordenadas2 = []
coorde3   = []
for i in range(total2):
    identif = coordenadas[i][6][1:13]
    octante = coordenadas[i][7][1:-2]
    patrone = coordenadas[i][8][1:-3]
    nombreI = str(identif)+str('_')+str(octante)+str(patrone)
    if any(nombreI in s for s in listaain1):
        coorde=np.insert(coordenadas[i],6,'AIN1')
        coorde2.append(coorde)
    elif any(nombreI in s for s in listaain2):
        coorde=np.insert(coordenadas[i],6,'AIN2')
        coorde2.append(coorde)
    elif any(nombreI in s for s in  listaain3):
        coorde=np.insert(coordenadas[i],6,'AIN3')
        coorde2.append(coorde)
    elif any(nombreI in s for s in listanorm):
        coorde=np.insert(coordenadas[i],6,'norm')
        coorde2.append(coorde)
    elif any(nombreI in s for s in listacond):
        coorde=np.insert(coordenadas[i],6,'cond')
        coorde2.append(coorde)
    elif any(nombreI in s for s in listanorB):
        coorde=np.insert(coordenadas[i],6,'norm')
        coorde2.append(coorde)
    else:
        coorde3.append(coordenadas[i]
coordenadas2 = np.asarray(coorde2)
```

Texto 6: Código: script para guardar en las listas de coordenadas el resultado de la biopsia

En las líneas 31,34,37,40,43 y 46 hay condicionales. Python permite buscar una cadena, nombreI en cualquier otra cadena *s* de entre todas las cadenas que hay en cada lista. Esto es otra muestra de la gran capacidad que Python ofrece , en una línea y dentro de una comprobación condicional permite iterar a lo largo de un bucle.

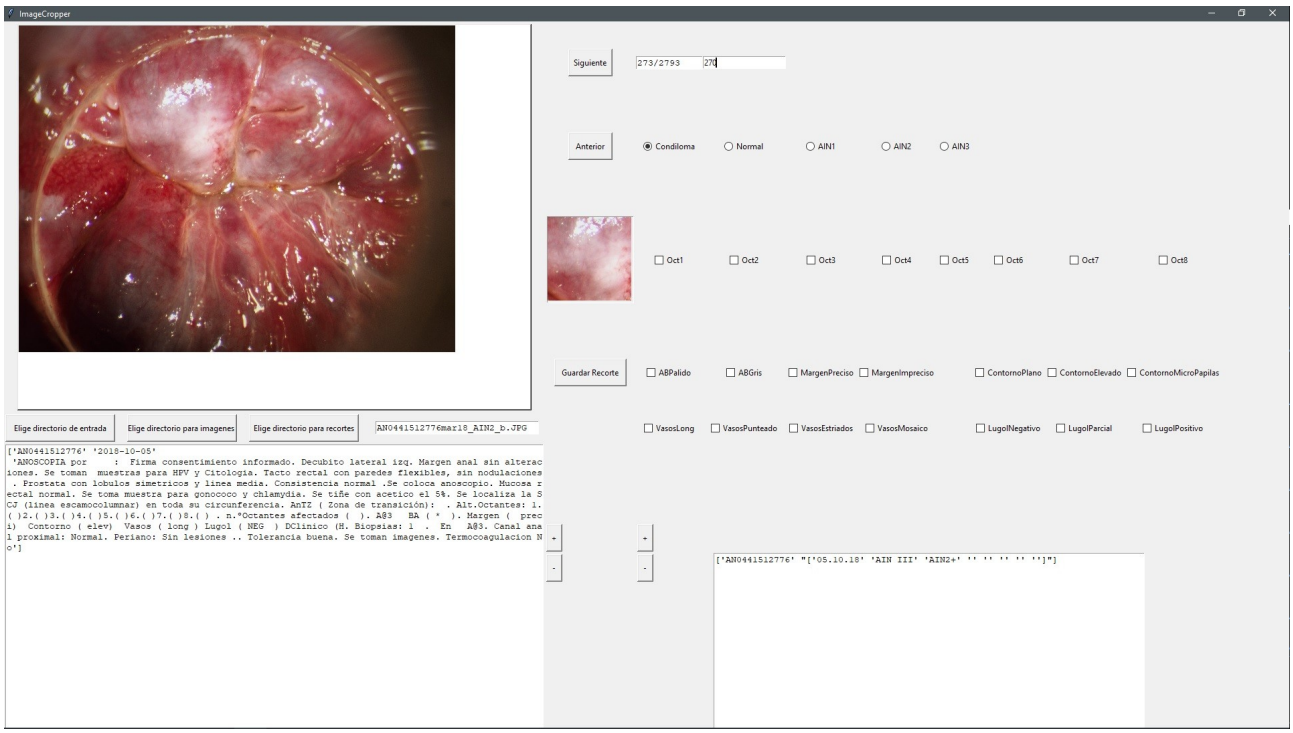


Figura 21: Aspecto de la aplicación ImageCropper, agregado los resultados de biopsias

Este paso intermedio podría haber sido evitado agregando al nombre del recorte el resultado de la biopsia. Esta modificación se realiza entonces a la aplicación ImageCropper. Se añaden, como puede verse en la figura 21 cinco selectores radiales. En contraposición con los selectores normales, éstos pueden estar todos sujetos al comportamiento de una misma variable, de forma que únicamente uno de todos los selectores podrá estar activo a la vez. Entonces, al activar alguno de ellos se hace una llamada a una función que modifica el valor de la variable cadenaNombreBiopsia con el resultado que proceda, además, cambia la ruta de destino de los recortes conforme a la estructura de carpetas que ha descrito previamente. En la función guardaCrop, al momento de generar el recorte y las coordenadas, se agrega el contenido de cadenaNombreBiopsia al nombre del recorte que se va a guardar, quedando así todo correctamente ordenado y etiquetado.

De esta forma se corrige el fallo. Si no hubiera sido necesario generar mas recortes, tampoco habría sido necesario realizar estas modificaciones, pero al leer la lista de coordenadas, se hacía imperativo conocer la clase de cada recorte para saber dónde guardarlo. Una vez se tienen todos etiquetados, no resulta estrictamente esencial mantener una estructura de directorios que separe las imágenes según su resultado patológico, porque cualquier sistema operativo tiene hoy en día eficientes funciones de búsqueda de archivos.

Durante la fase de entrenamiento de la red neuronal, se van a pasar a ésta una gran cantidad de imágenes de recortes. Para que el entrenamiento sea productivo, se ha de entregar también un vector que para cada recorte, tiene un identificador numérico que indica su clase. Esto entonces, se puede hacer cómodamente con las dos formas implementadas: encontrando la clase al descubrir la carpeta en la que el recorte ha sido guardado, o encontrar la clase en el nombre del recorte.

El script que se muestra en el cuadro de texto 7 es el utilizado para la extracción de recortes automática. Básicamente encuentra la posición x e y de la lista de coordenadas y calcula las posiciones x e y equivalentes en la imagen original, para poder trabajar con la mayor resolución posible y analizar así los resultados. Como las imágenes eran redimensionadas para su visualización, en las coordenadas están también los tamaños de las imágenes que fueron usadas para hacer los recortes, permitiendo así que con un cálculo básico como el de las líneas 30 y 31 se obtenga la posición para la imagen original.

```
coordenadas = np.genfromtxt('coordenadas1212_2.txt',dtype='str')
data_path = 'HPV_Imagen'
path_100  = '100m1_nuhsa'
path_200  = '200m1_nuhsa'
path_300  = '300m1_nuhsa'
save_path = 'recortes grandes2'
lista_100 = os.listdir(os.path.join(data_path,path_100))
lista_200 = os.listdir(os.path.join(data_path,path_200))
lista_300 = os.listdir(os.path.join(data_path,path_300))
#PARA ACORDARSE: EN COORDENADAS HAY LOS SIGUIENTES CAMPOS:
# POSX,POSY,TAMAÑO RECORTE,ALTURA IMAGEN,ANCHURA IMAGEN, CANALES,NOMBRES DE OCTANTE Y PATRON
# ES INCLUIDA LA CLASE ENTRE LOS CANALES Y LOS NOMBRES DE OCTANTE Y PATRON
total  = len(coordenadas)
#numant = 768
for i in range(round(total)):
    #print(coordenadas[i])
    img = coordenadas[i][7][1:-2]  #comienza despues y termina antes: quitar apostrofes
    posx = int(coordenadas[i][0][2:-1])
    posy = int(coordenadas[i][1][:-1])
    alto = int(coordenadas[i][3][:-1])
    anch = int(coordenadas[i][4][:-1])
    clase   = coordenadas[i][6]
    identif = coordenadas[i][7][1:13]
    octante = coordenadas[i][8][1:-2]
    patrone = coordenadas[i][9][1:-3]
    tam2=1500
    if (img in lista_100):
        print('100mil', img)
        original =
cv2.cvtColor(cv2.imread(os.path.join(data_path,path_100,img)),cv2.COLOR_BGR2RGB)
        posX = round(posx*original.shape[1]/anch)
        posY = round(posy*original.shape[0]/alto)
        crop = cropCrea(posX,posY,tam2,original)
        plt.imshow(Image.fromarray(crop))
        crop_save = cv2.cvtColor(crop,cv2.COLOR_RGB2BGR)
        cv2.imwrite(os.path.join(data_path,save_path,str(identif)+'_'+str(octante)
+'_'+str(patrone)+'_'+str(clase)+'_'+str(i+numant)+'.jpg'),crop_save)
        #extraer recorte de la imagen de la carpeta de los 100mil
    if (img in lista_200):
        print('200mil',img)
        original =
cv2.cvtColor(cv2.imread(os.path.join(data_path,path_200,img)),cv2.COLOR_BGR2RGB)
        posX = round(posx*original.shape[1]/anch)
        posY = round(posy*original.shape[0]/alto)
        crop = cropCrea(posX,posY,tam2,original)
        plt.imshow(Image.fromarray(crop))
        crop_save = cv2.cvtColor(crop,cv2.COLOR_RGB2BGR)
        cv2.imwrite(os.path.join(data_path,save_path,str(identif)+'_'+str(octante)
+'_'+str(patrone)+'_'+str(clase)+'_'+str(i+numant)+'.jpg'),crop_save)
        #extraer recorte de la imagen de la carpeta de los 200mil
    if (img in lista_300):
        print('300mil')
        #extraer recorte de la imagen de la carpeta de los 300mil
        #se procede como en los casos previos
```

Texto 7: Código: script para la generación automática de recortes desde los ficheros de coordenadas

Este script puede resultar de vital importancia para el entrenamiento de la red. La elección del tamaño de recorte del patrón es muy importante. Si se elige un tamaño muy pequeño, se omite información relativa a las estructuras adyacentes a la región de interés que podría hacer mas complicado el proceso de caracterizar las lesiones precancerígenas, el problema de los recortes pequeños es que se enfrentan a la posibilidad de no abarcar por completo la lesión, haciendo que la red experimente un proceso más difícil. Si los recortes tienen un tamaño grande, la lesión probablemente quedará bien contenida en el recorte, pero puede que otras estructuras también, forzando a la red a aprender a diferenciar elementos que pueden ser muy similares.

Como el tamaño por defecto que la aplicación ImageCropper produce es 128 por 128 píxeles extraídos de imágenes de entorno a los 400-500 píxeles por unos 300-400, se prueba con recortes de mayor tamaño. Las imágenes que el visor de la aplicación muestra son redimensionadas a unos 400-500 píxeles para poder verlas completas en pantallas pequeñas. Ya que las lesiones son complicadas de identificar, es necesario probar con varios diferentes. Además, el tamaño del recorte depende enormemente del tamaño de la imagen del que ha sido extraído, es decir, una imagen de gran tamaño de la que se recorta una zona pequeña probablemente no permita ver toda la lesión entera. Si se extrae una zona mediana de una imagen grande, es probable que sí se encierre una lesión en su totalidad y además, existe la posibilidad de redimensionar *a posteriori* los recortes. Esto haría que recortes en los que se pueda ver bien la lesión sigan teniendo un tamaño reducido para reducir la carga computacional que supone el entrenamiento de la red.

Todos estos parámetros deben mantenerse bajo control para realmente controlar el progreso del entrenamiento. En el texto 7, en la línea 26 se puede modificar el tamaño del lado del cuadrado que se extrae de la imagen original. En este momento también es posible modificar la forma del recorte que se extrae, en la línea 32 se llama a la función cropCrea, del texto 5, dentro de la cual, como se comenta, puede seleccionarse una zona rectangular, circular o de la forma deseada.

5.6. Preprocesamiento de las imágenes

El preprocesamiento de las imágenes representa una parte muy importante del trabajo. Con técnicas de preprocesamiento adecuado, las características deseadas pueden ser mas sencillas de detectar es decir, se puede condensar información y filtrar ruido para poder centrar el estudio en aquello que realmente interesa. A la hora de alimentar con datos una red neuronal, el preprocesamiento puede hacer que pocos datos sean suficientes para caracterizar un comportamiento.

5.6.1 Eliminación de brillos

La eliminación de brillos puede ser una buena ayuda. Las imágenes de HRA tienen muchas pequeñas zonas completamente saturadas porque el uso de lubricante crea gran cantidad de reflejos en los tejidos. Para los médicos, corregir los brillos puede ayudar a ver con claridad la imagen, razón por la cual se decide a buscar un método.

Con una umbralización sencilla se puede generar rápidamente una máscara de la imagen en la que se seleccionan los puntos totalmente saturados. Seguidamente se puede operar seleccionando individualmente cada mancha saturada de blanco. Una vez se centra el procesado en una zona de brillos, se pueden realizar operaciones morfológicas. De esa forma, es posible que automáticamente los valores que rodean los puntos de brillos se cierren sobre las zonas saturadas. El principal problema que esto plantea es evidente, esos valores son ficticios. En casos en los que los puntos brillantes son muy pequeños, existe una alta probabilidad de que los valores que se encontraban bajo los brillos sean muy parecidos a los de los píxeles adyacentes. Sin embargo, en las zonas saturadas de mayor área se llega a notar el procesado si no es suficientemente cuidadoso, ya que si es necesario iterar varias veces para cerrar un punto, se nota que ese punto deja de ser blanco y saturado para tomar valores homogéneos. Los puntos se rellenan con los valores de los límites, si son necesarias varias pasadas, esos valores se repiten, creando un rastro visible pero que los médicos consideran aceptable frente a los brillos.

Con opencv se puede aprovechar la capacidad de la función *inpaint*. Inpaint realiza una reconstrucción de un área seleccionada de una imagen desde los píxeles que están al borde de dicho área. Partiendo de los valores frontera del área señalada, el algoritmo trata de rellenar pequeños vecindarios en torno a cada píxel que se procesa en la región de interés. Basar los valores de la reconstrucción en los valores vecinos pretende dar coherencia a la transformación. El nombre surge de image interpolation, que da la idea clave de qué hace la función. En la version de inpaint que ofrece opencv se debe facilitar a la función la imagen o zona de la imagen que quiera reconstruirse, una máscara que envuelva la zona de interés sobre la que se desea que actúe el algoritmo, se puede facilitar

```
1.  #imagen  = cv2.cvtColor(imagen,cv2.COLOR_BGR2RGB)
2.  thresh  = 0.95
3.  imUmbral= imagen/255 > thresh
4.  kernel = np.ones((50,50),'uint8')
5.  imUmbraldilate = cv2.dilate(np.uint8(imUmbral),kernel)
6.  im_mask_1d=imUmbraldilate[:,:,1]
7.  imUmbralInpainting = cv2.inpaint(imagen,im_mask_1d,1,cv2.INPAINT_TELEA)
8.  plt.figure(figsize=(30,10))
9.  plt.subplot(2,2,1)
10. plt.imshow(imUmbral*255)
11. plt.subplot(2,2,2)
12. plt.imshow(np.uint8(imagen-imUmbral*255))
13. plt.subplot(2,2,3)
14. plt.imshow(imUmbraldilate[:,:,1])
15. plt.subplot(2,2,4)
16. plt.imshow(imUmbralInpainting)
17. plt.savefig('resultado_brillos_umbral.jpg')
```

Texto 8: Código: script que corrige los brillos de la imagen

además el radio de acción así como especificar el método que gobernará el comportamiento del algoritmo. Esta función resulta bastante eficaz y procesa rápidamente imágenes de gran tamaño, razones por las cuales es elegida.

Para generar la máscara de actuación se recurre a un método mas simple, el umbralizado. Con ese ajuste se puede retener únicamente la información de una imagen que toma valores específicos indicados por el valor del umbral. Así, si se quiere retener únicamente las zonas saturadas, se debe poner un alto umbral porque los brillos y la alta saturación en la imagen suelen tomar los valores más altos de entre los que el rango dinámico ofrece. En el bloque de código 8 puede verse que el umbral se establece en 0.95. Aunque es un valor próximo al límite superior, es posible que algunas zonas como se ve en la figura 22 imagen a) también superen el umbral por estar fuertemente saturadas. De esa imagen resultado se puede seleccionar únicamente un canal, en este caso se prueba a no elegir el rojo, puesto que en las imágenes de HRA se prevé que habrá con frecuencia zonas rojas muy saturadas, y como puede verse en la imagen b) se podría acabar actuando sobre zonas que no son brillos.

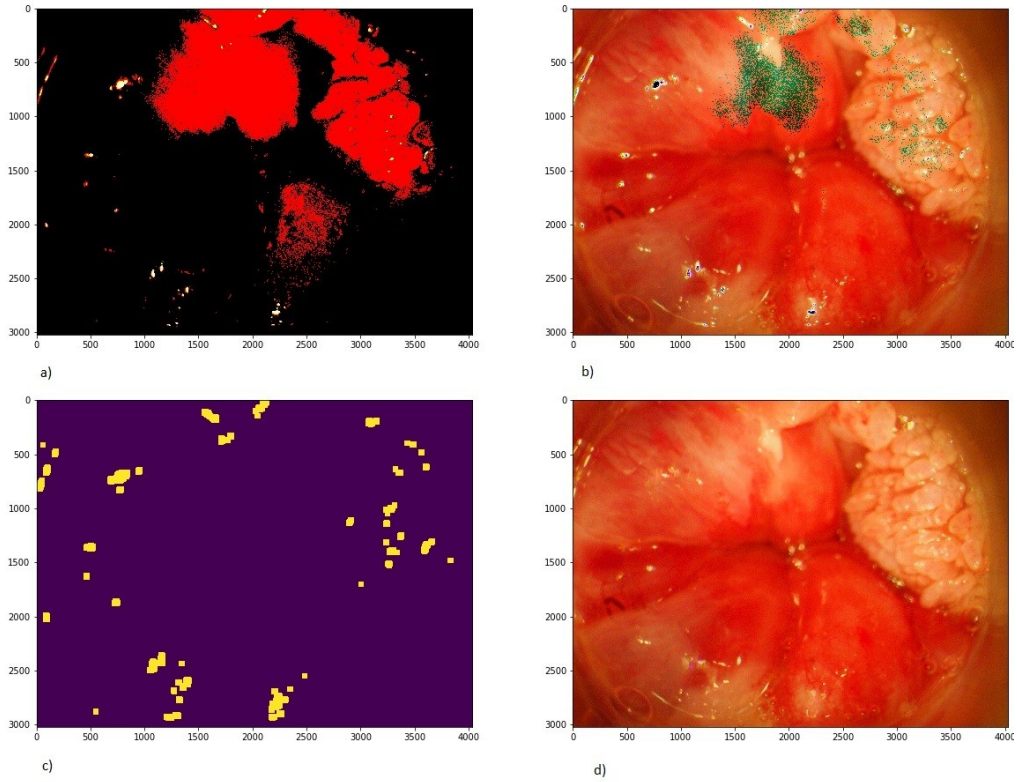


Figura 22: Resultado de las fases de retirado de brillos. De izquierda a derecha y de arriba a abajo: a) resultado del umbralizado, se ve que se mantiene información de los tres planos de color, puede notarse cómo hay zonas casi totalmente saturadas de rojo, b) imagen original con el umbralizado superpuesto para observar las zonas concretas que han superado el umbral, c) máscara para pasar a la función inpaint, ha sido dilatada y se ha seleccionado únicamente un canal, d) resultado final de la reducción de brillos

5.6.2 Detección del círculo del borde del anoscopio

La detección del borde del anoscopio también resulta útil, porque se puede centrar la imagen solamente en la zona de interés. La detección sin embargo conlleva algunas dificultades: a veces no está correctamente centrado en la imagen, otras veces no se tiene el borde representado en su totalidad, otro problema es que los detectores de bordes genéricos no dan buen resultado sin aplicar trabajo extra porque las imágenes de HRA tienen muchos

elementos y grandes cantidades de zonas con altas frecuencias que son susceptibles de confundir a los detectores genéricos.

La correcta detección del borde sin embargo permite una funcionalidad extra: con el centro bien situado y el diámetro correcto, se podría realizar la extracción basada en las Hojas de Evolución que tengan bien especificada la posición de toma de biopsias. Como se ha visto anteriormente, las mencionadas hojas tienen una descripción de la intervención que el profesional realiza. Normalmente indican la posición ya sea en octantes u horas del reloj en la que toman la biopsia, entonces si se tiene constancia del círculo del anoscopio, se puede intentar realizar la extracción automatizada. Esto sin embargo se enfrenta a varios problemas: la posición de toma de biopsias se especifica de forma variable durante todo el tiempo que dura la primera fase del proyecto POMPIS, así que es necesario primero filtrar las hojas que no especifican bien la localización de la biopsia. Otro inconveniente es la diferencia entre las resoluciones de las imágenes, que harán que igualar los recortes en tamaño para su procesado posterior sea menos sencillo que trabajar con recortes cuadrados. Las imágenes digitales son guardadas en matrices que tienen formas cuadradas o rectangulares así que tomar recortes con forma de sección circular conllevaría rellenar de negro las zonas que sobran del mínimo rectángulo que envuelva el sector. Esto finalmente no queda dentro del desarrollo de este trabajo de fin de grado ya que en última instancia, si se entrena una red con recortes cuadrados, luego puede clasificarse una imagen entera si ésta se divide en recortes cuadrados que van solapándose y pasando por la red neuronal. De esta forma, se podría representar el resultado de la clasificación sobre la imagen original haciendo el trabajo bastante independiente de la forma del recorte. Ver tabla 5.

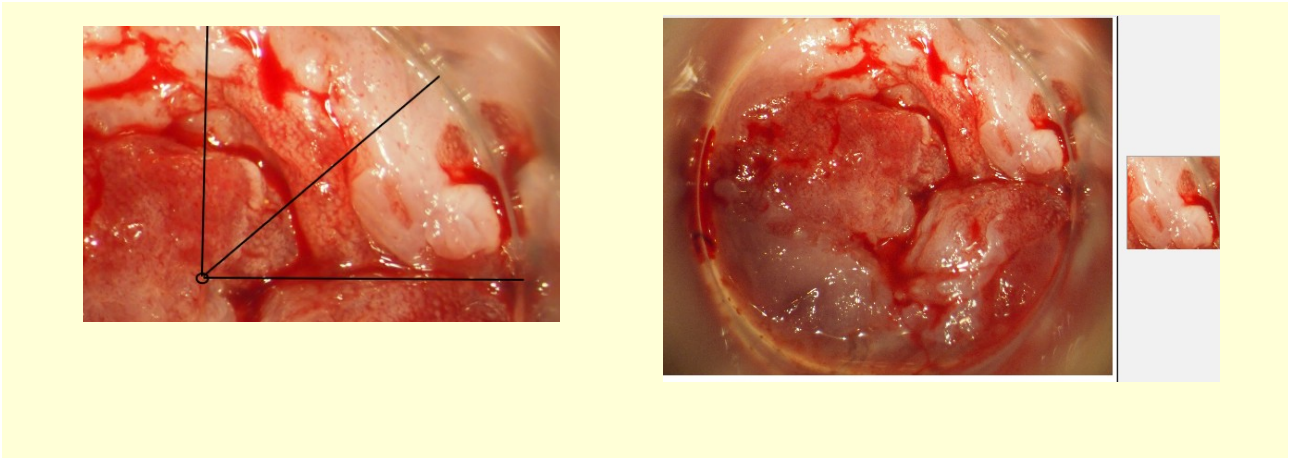


Tabla 5: Comparación de recorte de octante (izquierda) y recorte cuadrado (derecha)

A pesar de lo comentado en el párrafo superior, se prueban algunas formas básicas de detectar el círculo: haciendo un filtrado de bordes con una máscara de Sobel y luego procesando la imagen resultado con un detector de círculos de Hough. Este último paso se puede realizar con la función de opencv HoughCircles. Éste método tiene el problema de no ser muy veloz además de que a menudo falla encontrando exactamente el centro y el diámetro de la imagen. Otro método que se prueba es asistido y resulta rápido en su procesamiento, pero requiere que un usuario seleccione tres puntos que pertenecen al borde del círculo. En este caso la detección es tan buena como el usuario se proponga, pero entre tantas imágenes resulta excesivo procesarlas todas a mano y de una en una.

En las figuras 23 y 24 se muestran algunos ejemplos en los que la detección resulta compleja, puede verse cómo los algoritmos tienen un acierto discutible. Para hacer énfasis se muestra la segunda mejor opción de círculo que envuelve al anoscopio que se detecta, y puede verse que en ambas figuras 23 y 24 otro objeto es declarado como círculo aunque es evidente que no es así. La imagen procesada es en ambos casos el resultado de un filtrado de Sobel de las originales.

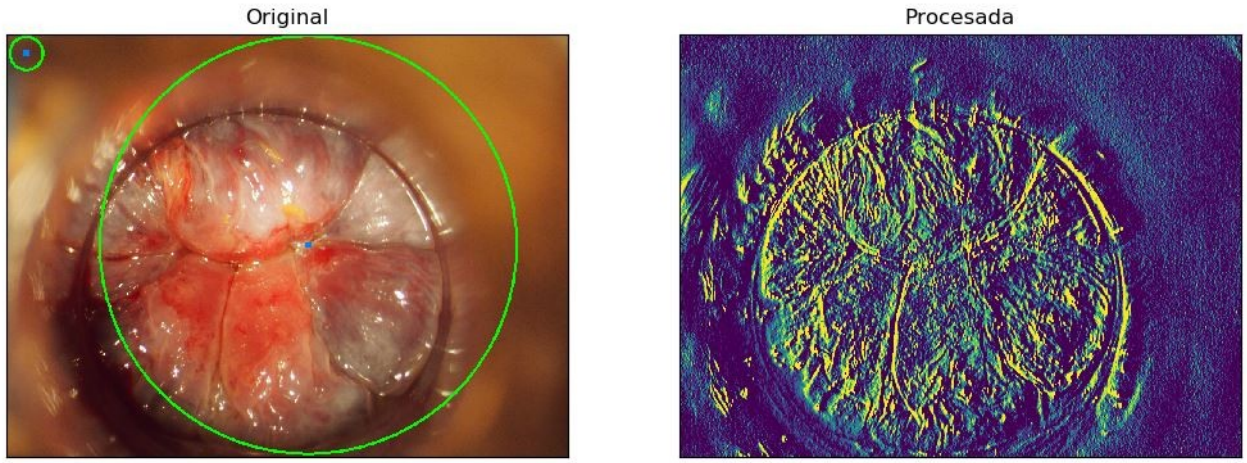


Figura 23: Detección de círculo del anoscopio incorrecta

En la figura 23 se puede ver la dificultad que tiene el algoritmo para encontrar correctamente el borde inferior e inferior izquierdo, como es la zona del anoscopio que queda mas oscura, se crea un puente suave de bajas frecuencias que hacen que el filtro de Sobel pase por alto un borde tan ancho. Esta generación de sombras puede afectar y es impredecible, pues dependerá de cómo se está iluminando la zona de interés por los profesionales durante los análisis clínicos.

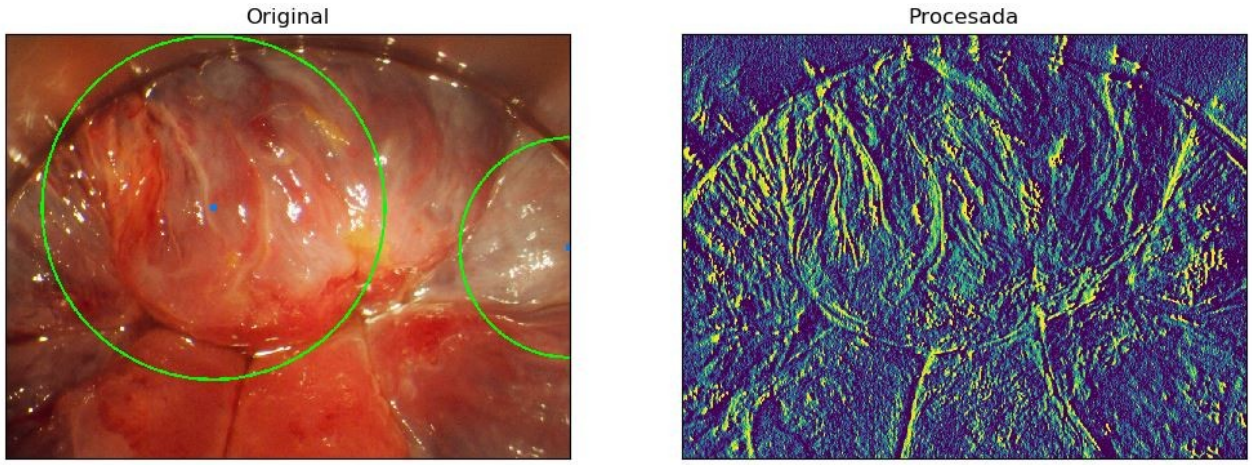


Figura 24: Detección incorrecta del círculo del anoscopio

La figura 15 muestra otro ejemplo claro: las imágenes en las que los médicos buscan un detalle mayor representan una dificultad de los algoritmos para encontrar el círculo completo con solamente una sección disponible.

Una propuesta para encontrar con mayor acierto el círculo sería realizar un algoritmo basado en contornos activos en el que la función trate de encontrar la mayor diferencia entre la zona exterior del anoscopio y el interior ya que a simple vista se ve que el exterior está muy difuminado y el interior suele tener gran contenido de altas frecuencias, así que podría buscarse la curva que mejor separa dichas áreas. Otra propuesta sería entrenar una red neuronal para segmentar el borde, en caso de que resultara mas complicado de lo esperado, de forma que se generase una máscara con la zona interior del anoscopio seleccionada.

Finalmente se muestran dos ejemplos del borde correctamente detectado y con la división en octantes superpuesta.

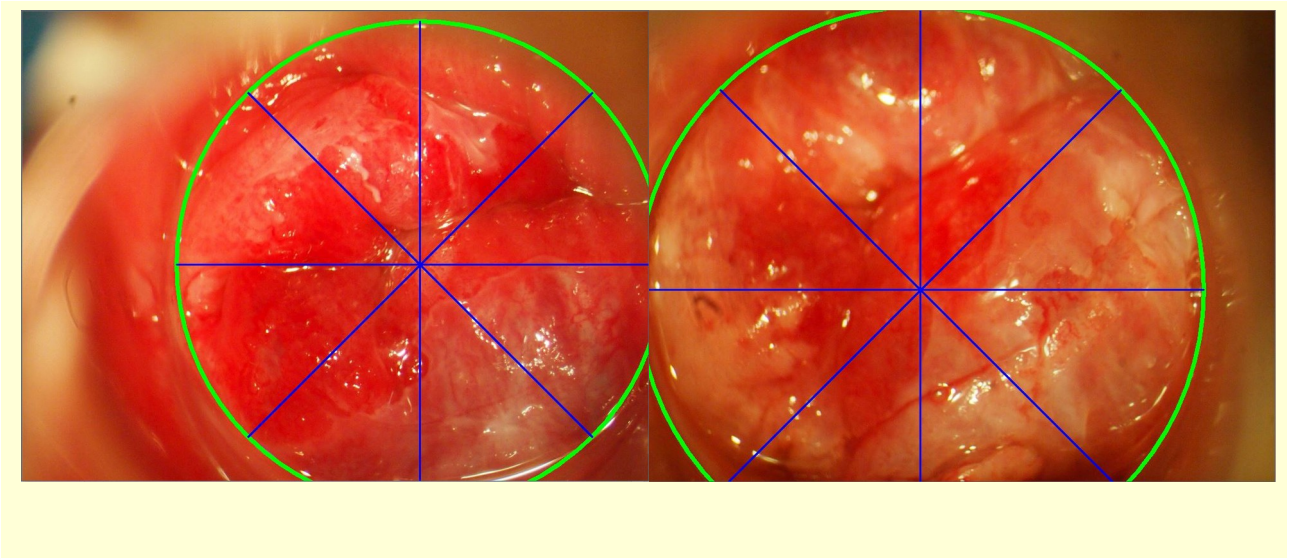


Tabla 6: Ejemplos de imágenes con el borde del anoscopio detectado y superposición de los octantes

5.6.3 Mapas auto-organizados SOM

En un caso de estudio similar, pero relativo al cáncer de cérvix [2] se utilizan redes auto-organizadas de Kohonen para procesar las imágenes e intentar conseguir una mejora en la clasificación.

Los mapas auto organizados (Kohonen 1990), crean una representación organizada de varias características de las señales, en este caso imágenes. Son especialmente útiles en el campo del reconocimiento de patrones. Los mapas auto-organizados son un tipo de red neuronal llamado competitivo. Se trabaja intentando generar grupos o clusters de datos que se ciernen en torno a un patrón. Cada neurona o grupo de neuronas actúa como un decodificador separado para cada entrada y es la respuesta activa en esa localización la que resulta en una interpretación de la información de entrada.

A la vista, el resultado de los mapas auto-organizados recuerda al análisis de componentes principales o PCA. Sin embargo, el mapa auto-organizados se genera con un algoritmo que tiene una estructura de red neuronal, la PCA se basa en la búsqueda de los autovalores y autovectores que contienen mayor cantidad de información.

La clave es que los mapas de Kohonen organizan la segregación espacial en subconjuntos relacionados topológicamente. Los SOM permiten representar datos que parten de un origen de muchas dimensiones en soportes de menor número de dimensiones. Finalmente se tiene una representación discretizada o cuantizada de los datos de entrada. Estos mapas no necesitan un resultado objetivo con el que minimizar la diferencia, en lugar de eso, se trata de optimizar el parecido entre los vectores de pesos de las neuronas y el vector de entrada. Los grupos de puntos en los que la entrada y los pesos son parecidos se separarán en zonas diferentes del mapa

Python tiene a disposición del usuario el uso de la librería MiniSom para el desarrollo y empleo de las técnicas de mapas auto-organizados. Está pensada y basada en el uso con numpy, así que deben organizarse los datos en matrices entendibles por la librería numpy.

El proceso de entrenamiento es sencillo, para entradas x se tienen pesos w:

1. Se inicializan los pesos de las neuronas. Existe en MiniSom el método train_random que permite tomar valores de pesos aleatorios. Se inicializa el tiempo a cero

2. Se toma una entrada aleatoria x del conjunto de datos

3. Se elige la neurona ganadora *i* como aquella que minimiza la diferencia entre la entrada y el peso *i*, reciben el nombre de BMU (por sus siglas en inglés, Best Matching Unit) y se emplea la distancia euclídea:

$$argmin_i \, ||x-w_i||$$

4. Tras determinar la BMU, se buscan las otras neuronas o nodos que pertenecen a su vecindario. Para ello se utiliza una función de vecindad que puede ser por ejemplo un hexágono, un cuadrado o una función gaussiana centrada en la posición de la BMU.

5. Adaptar los pesos de cada nodo según la función expresada abajo. Donde $\eta(t)$ es el ratio de aprendizaje, que es una función decreciente con el tiempo, por ejemplo una exponencial y $h(t)$ es la función de vecindad, que tiene a valor alto la BMU y las neuronas que pertenecen a su vecindario y a valor bajo las neuronas que no. Así, en cada iteración las neuronas adyacentes a la BMU ajustan sus pesos para responder mejor al patrón.

$$w_n = w_n + \eta(t) \cdot h(i) \cdot (x - w_n)$$

6. Se incrementa el tiempo.

El mapeo de características en imágenes con las técnicas de mapas auto-organizados resulta eficiente con imágenes de pequeña resolución, pero es evidentemente muy costoso de realizar en imágenes grandes. Se muestran algunos ejemplos a continuación.

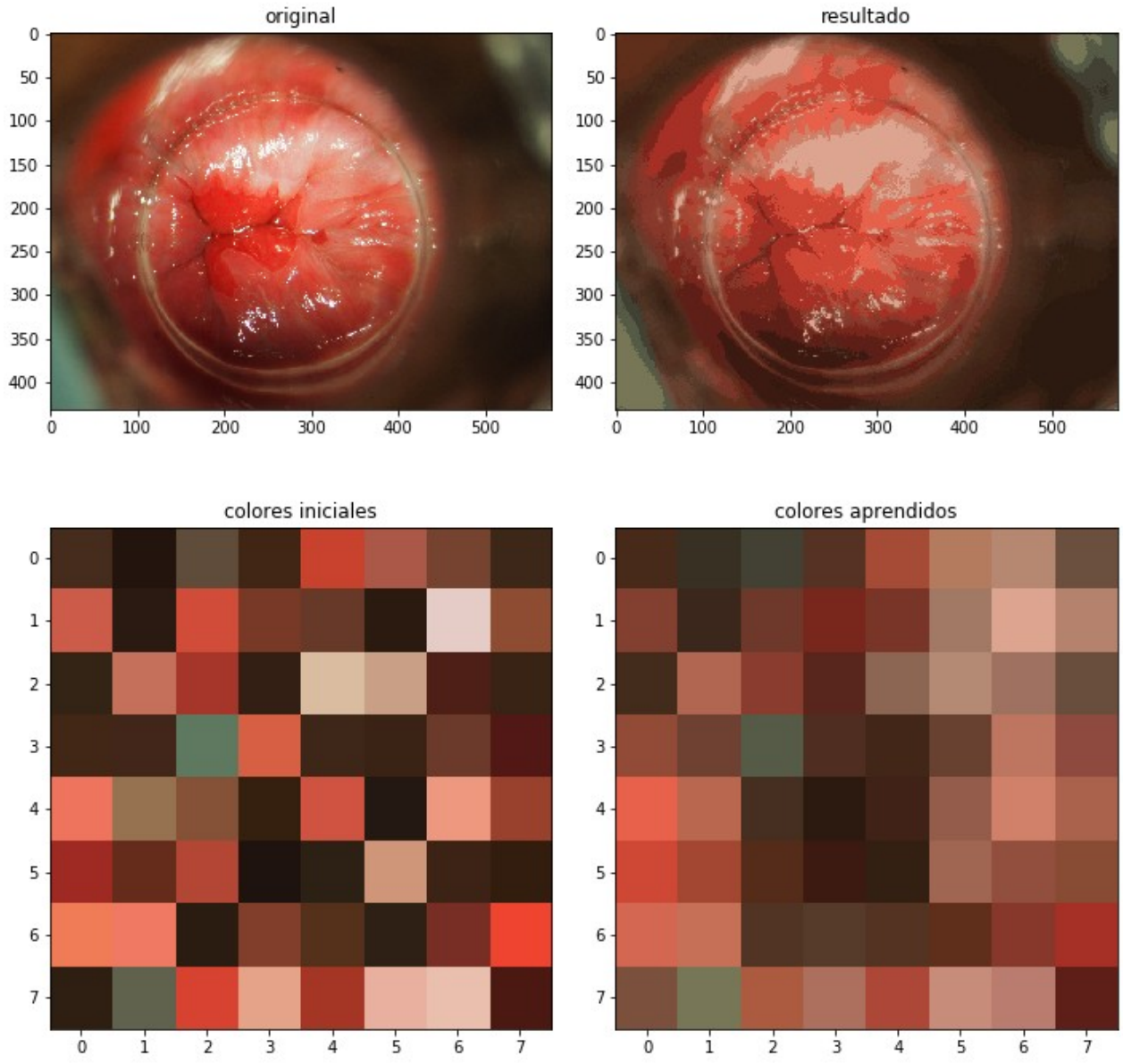


Figura 25: Resultado del entrenamiento del mapa auto-organizado

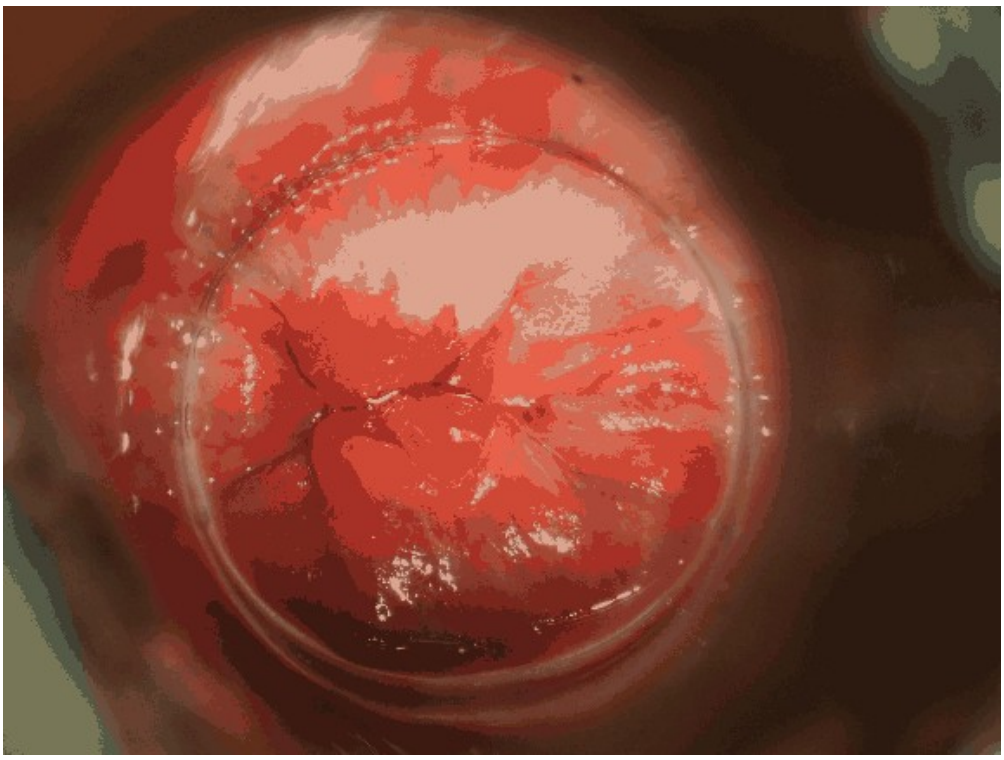


Figura 26: Mapa auto-organizado de una imagen de HRA

5.6.4 Preprocesamiento con keras. Normalización y data augmentation

Keras suministra una función que permite normalizar datos. La función acepta una matriz y retorna esa matriz normalizada según una norma que puede especificarse. Permite por ejemplo usar una norma L2, Euclídea o como termina siendo usada en este trabajo, imponiendo un ajuste de valores de píxel entre 0 y 1 en los datos. Este paso es necesario porque las imágenes se codifican como enteros sin signo. Estos enteros son normalmente codificados en 8 bits, generando 256 valores que suelen ir desde el 0 como el tono mas oscuro o negro hasta el 255 que suele ser el valor mas alto de saturación o blanco. Si se mantuviera la codificación, las operaciones relativas al trabajo con los valores de los píxeles resulta mas cómoda.

De la mano de la normalización viene la estandarización. En este caso se trata de realizar un escalado de los datos asumiendo que la distribución de los valores de los píxeles es gaussiana. Se realiza un escalado de dichos valores para forzar a que su media sea 0 y su varianza 1. Esto se consigue restando la media y dividiendo entre la varianza de los datos.

Keras dispone de una herramienta tremendamente útil que se llama *ImageDataGenerator*. Con esta función es posible indicar toda una batería de transformaciones que las imágenes experimentarán al entrenar la red. De esta forma lo que la función produce es un generador de transformaciones y así, al hacer pasar el conjunto de datos a través de la función, las imágenes se van transformando de manera aleatoria y eso permite que en cada vuelta que el entrenamiento da a los datos las transformaciones no sean las mismas en cada imagen.

Otra gran ventaja del uso de este tipo de preprocesamiento es que no se necesita espacio físico de almacenamiento para guardar todas esas imágenes, aunque ello es perfectamente viable, también resulta provechoso hacer esas transformaciones para que sea únicamente la memoria temporal la que las almacene. Evidentemente, para conjuntos de datos de gran calidad y resolución, puede ser provechoso el uso de estas técnicas.

Keras ofrece de entre muchas otras opciones las siguientes para realizar transformaciones en las imágenes que se van a tratar (para ver un ejemplo de uso ir a texto 11) :

- Normalización estándar de los datos, dividiendo las entradas entre la desviación típica de la entrada actual o de todas las cargadas.

- Rango de rotaciones, *ImageDataGenerator* permite que se indique un rango en grados para girar las imágenes de forma aleatoria en ese rango.

- Desplazamientos verticales y horizontales, también se puede indicar un rango de píxeles en el que las imágenes serán desplazadas. En este y en otros casos es necesario hacer *zero padding* para mantener el tamaño original de la imagen. Esto quiere decir que si se desplaza la imagen en una dirección, parte de la imagen se perderá porque sale de la visión y otra parte debe ser agregada, esto se hace añadiendo ceros a la imagen cosa que añade franjas negras. En este caso, se prefiere no realizar este tipo de transformaciones para no agregar dificultades a la red y porque además existen varios recortes que son tomados cerca unos de los otros y suplen ya la necesidad que resuelve este tipo de transformaciones.

- Rango de brillo, se puede alterar el brillo medio de la imagen a uno establecido dentro de un rango que debe indicarse con dos valores enteros.

- Volteo vertical y horizontal, es posible hacer una imagen especular en torno al eje horizontal o vertical para conseguir otra transformación sencilla.

El uso del *data augmentation* de Keras permite generar varias imágenes de cada fotografía del conjunto de datos. Esto permite crear 5 copias alteradas de cada original. Así, se tiene una herramienta tremendamente poderosa para aprovechar aun más los datos disponibles pero, incurrir en un proceso de *data augmentation* demasiado severo puede llevar a conclusiones y datos engañosos. Aunque se haga una transformación, las imágenes mantienen un enorme parecido con las originales,y los patrones que existen, se ven poco alterados pero, no hay duda de que hace la detección independiente de la posición y orientación de los tejidos que se están tratando de identificar.

5.7. Preparación de los conjuntos de entrenamiento y validación

En este punto se dispone de recortes de patrones clasificados en resultados de biopsias. Las clases son: condiloma, normal, displasia de grado bajo o AIN1, displasia de grado medio o AIN2 y displasia de grado alto o AIN3. Existe otra clasificación posible, lesiones displásicas y no displásicas. En el primer grupo están las últimas tres clases antes mencionadas y en el segundo, los patrones normales y los condilomas puesto que son los casos en los que no hay lesión precancerígena.

Para entrenar una red neuronal y que logre clasificar es necesario proveer de duplas compuestas de una muestra de datos y la clase a la que pertenecen. En este caso la muestra de datos es una matriz que representa una imagen y para la clase se utiliza un entero positivo.

```
1.  dirEntrena  = 'HPV_Imagen/train/'
2.  image_rows  = 1500
3.  image_cols  = 1500
4.  dirEntrenaDisp = os.path.join(dirEntrena,'Displasia')
5.  dirEntrenaNorm = os.path.join(dirEntrena,'Norm')
6.  entrenaNorm = os.listdir(dirEntrenaNorm)
7.  entrenaDisp = os.listdir(dirEntrenaDisp)
8.  imgsTrain   = entrenaNorm + entrenaDisp
9.  total1      = len(imgsTrain)
10. totalNorm   = len(entrenaNorm)
11. totalDisp   = len(entrenaDisp)
12. ancho = round(image_rows/10)
13. alto  = round(image_rows/10)
14. dim = (ancho,alto)
15. print('Total de imagenes de entrenamiento:',total1)
16. imagenes = np.ndarray((total1,alto, ancho,3), dtype=np.uint8)
17. #intentaremos generar los vectores de verdades de referencia, las imágenes están mezcladas
18. random.shuffle(imgsTrain)
19. coincide1 =np.where(np.in1d(imgsTrain,entrenaDisp))[0]
20. coincide2 =np.where(np.in1d(imgsTrain,entrenaNorm))[0]
21. Y_train   =np.ndarray((total1), dtype=np.int)
22. print(imgsTrain[12])
23. for p in tqdm(range(total1)):
24.     if(p in coincide1):
25.         img =
cv2.cvtColor(cv2.imread(os.path.join(dirEntrenaDisp,imgsTrain[p]),1),cv2.COLOR_BGR2RGB)
26.         Y_train[p]  = 1
27.     if(p in coincide2):
28.         img =
cv2.cvtColor(cv2.imread(os.path.join(dirEntrenaNorm,imgsTrain[p]),1),cv2.COLOR_BGR2RGB)
29.         Y_train[p]  = 0
30.     imagenes[p]=img
31. print('Carga de imagenes completada')
```

Texto 9: Código: script que realiza la generación del set de entrenamiento (train set) incluido el vector de clases

```
dirTest  = 'HPV_Imagen/test/'
dirTestAIN3 = os.path.join(dirTest,'AIN3')
dirTestAIN2 = os.path.join(dirTest,'AIN2')
dirTestAIN1 = os.path.join(dirTest,'AIN1')
dirTestNorm = os.path.join(dirTest,'Norm')
dirTestCond = os.path.join(dirTest,'Cond')
testeaAIN3 = os.listdir(dirTestAIN3)
testeaAIN2 = os.listdir(dirTestAIN2)
testeaAIN1 = os.listdir(dirTestAIN1)
testeaNorm = os.listdir(dirTestNorm)
testeaCond = os.listdir(dirTestCond)
imgsTest   = testeaAIN3 + testeaAIN2 + testeaAIN1 + testeaNorm + testeaCond
total1      = len(imgsTest)
print('Total de imagenes de test:',total1)
imagenes2 = np.ndarray((total1,image_rows, image_cols,3), dtype=np.uint8)
#intentaremos generar los vectores de verdades de referencia, las imágenes están mezcladas
random.shuffle(imgsTest)
coincide1 =np.where(np.in1d(imgsTest,testeaAIN3))[0]
coincide2 =np.where(np.in1d(imgsTest,testeaAIN2))[0]
coincide3 =np.where(np.in1d(imgsTest,testeaAIN1))[0]
coincide4 =np.where(np.in1d(imgsTest,testeaNorm))[0]
coincide5 =np.where(np.in1d(imgsTest,testeaCond))[0]
Y_test   =np.ndarray((total1), dtype=np.int)
print(imgsTest[12])
for p in tqdm(range(total1)):
    if(p in coincide1):
        img =
cv2.cvtColor(cv2.imread(os.path.join(dirTestAIN3,imgsTest[p]),1),cv2.COLOR_BGR2RGB)
        Y_test[p]  = 4
    if(p in coincide2):
        img =
cv2.cvtColor(cv2.imread(os.path.join(dirTestAIN2,imgsTest[p]),1),cv2.COLOR_BGR2RGB)
        Y_test[p]  = 3
    if(p in coincide3):
        img =
cv2.cvtColor(cv2.imread(os.path.join(dirTestAIN1,imgsTest[p]),1),cv2.COLOR_BGR2RGB)
        Y_test[p]  = 2
    if(p in coincide4):
        img =
cv2.cvtColor(cv2.imread(os.path.join(dirTestNorm,imgsTest[p]),1),cv2.COLOR_BGR2RGB)
        Y_test[p]  = 1
    if(p in coincide5):
        img =
cv2.cvtColor(cv2.imread(os.path.join(dirTestCond,imgsTest[p]),1),cv2.COLOR_BGR2RGB)
        Y_test[p]  = 0
    imagenes2[p]=img
print('Carga de imagenes completada')
```

Texto 10: Código: script que realiza la generación del conjunto de validación (test set) incluido el vector de clases

Keras pone a disposición del usuario funciones que simplifican el proceso descrito anteriormente. Estas funciones son *flow* y *flow_from_directory*. Con ellas, se puede generar un vector de clases que acompañe al grupo de imágenes. Siguiendo la estructura de directorios que antes se ha descrito: se debe tener una carpeta de

entrenamiento y otra de validación, dentro de las cuales tendremos otro directorio para cada clase, en el cual estarán todos los datos que pertenezcan a cada clase. Las herramientas de Keras pasan por cada directorio y asignan a esos datos la clase y luego pasan al siguiente directorio. Se debe indicar la ruta de acceso, el tamaño objetivo de los datos, en caso de ser imágenes se debe indicar si son en blanco y negro o en color, se puede indicar que se mezclen o no las imágenes así como la variedad de clases, si será *binary* con únicamente dos clases a predecir o *categorical* en caso de tener mas de dos.

Con estos scripts se tiene cierta comodidad para resolver los pasos previos al entrenamiento. Existe la posibilidad de redimensionar cómodamente todos los recortes, que es una cuestión que puede facilitar las pruebas rápidas así como realizar preprocesado o aumentado de datos justo antes de realizar los entrenamientos, dando la posibilidad de evitar tener que guardar todas las imágenes a las que se le realice *data augmentation*, pudiendo directamente entrenar con ellas.

El problema que se tiene al usar los scripts es que cargar todas las imágenes a la vez en memoria es muy costoso. La alternativa de Keras resulta cómoda porque se va cargando en la memoria del computador la cantidad de datos que se especifique y se va iterando sobre ellos para que, una vez procesados se retiren de la memoria y se carguen otros nuevos. Este detalle es fundamental ya que la alternativa a esto es usar scripts como los diseñados y mostrados en los textos 9 y 10 con la salvedad de que habría que ir realizando entrenamientos de pequeños grupos de datos, guardar el estado de la red (los pesos), cargar otro paquete de datos, cargar los pesos de nuevo y continuar entrenando. Además, en este caso, apenas hay 5 clases, pero si se debiera realizar esa estructura de condicionales dentro de un bucle para 1000 clases, resulta mucho mas eficaz el uso de scripts generalistas, puesto que es sencillo tener junto con el resultado del etiquetado de las funciones automatizadas una referencia de la clase real a la que pertenecen.

El uso de *flow* y *flow_from_directory* es muy cómodo si además se aprovecha para aplicar esas funcionalidades junto con el punto anterior. Es posible además en esos momentos seleccionar un tamaño para redimensionar todas las imágenes del conjunto de datos sin de nuevo, tener que guardar las imágenes de dimensiones diferentes a las originales.

```
1.  aug = ImageDataGenerator(rotation_range=20,rescale=1./255,
2.                           horizontal_flip=True,vertical_flip=True)
3.  train_generator = aug.flow_from_directory(
4.                       '/content/drive/My Drive/datos_entrenamiento/train',
5.                       target_size=(1000,1000),
6.                       batch_size=32,
7.                       class_mode='binary')
8.  validation_generator = aug.flow_from_directory(
9.                       '/content/drive/My Drive/datos_entrenamiento/test',
10.                      target_size=(1000,1000),
11.                      batch_size=32,
12.                      class_mode='binary')
13. H=model.fit_generator(train_generator,
14.                       steps_per_epoch=49,shuffle=True,
15.                       epochs=1,
16.                       validation_data=validation_generator,
17.                       validation_steps=4)
```

Texto 11: Código: ejemplo de uso de ImageDataGenerator y flow_from_directory

5.8. Pasos previos al entrenamiento. Pruebas

Antes del desarrollo del presente trabajo, se intentan algunas cuestiones previamente para ir entendiendo la dinámica de trabajo con las herramientas de *deep learning*.

5.8.1 Base de datos MNIST. Clasificación

El primer ejercicio es un "hola mundo" del *deep learning* y consiste en realizar un clasificador para la base de datos MNIST. Ésta contiene 60000 imágenes de entrenamiento y 10000 imágenes de validación todas ellas imágenes de dígitos escritos a mano. Se encuentran normalizadas y codificadas en escala de grises, además están ajustadas a un reducido tamaño de 28x28 píxeles.

Esta base de datos ha sido objeto de múltiples pruebas por parte de la comunidad, siendo alcanzado un ratio de error de hasta un 0,21% lo cual implica un nivel de precisión comparable al de un ser humano.

Debido a que es un ejemplo sencillo, las imágenes tienen cada una una cantidad de 784 datos (28^2) ya que únicamente tienen un canal por estar en escala de grises. Además esos datos son enteros sin signo codificados a 8 bits. Todo esto permite que como primer ejemplo se intente desarrollar una red de tipo MLP o *fully connected*. Sucede que al trabajar con grupos de datos manejables, se puede realizar con la red una estructura que conecte todas las neuronas de una capa con cada una de las neuronas de la siguiente capa.

Este tipo de modelos es muy sencillo de implementar y con el uso de librerías específicas como Keras más aún. A continuación se muestra un ejemplo de la arquitectura del modelo usado para clasificar la base de datos MNIST.

```
1. model = tf.keras.models.Sequential()
2. model.add(tf.keras.layers.Dense(128,activation=tf.nn.relu, input_shape = x_train.shape[1:]))
3. model.add(tf.keras.layers.Dense(128,activation=tf.nn.relu))
4. model.add(Dense(10,activation=tf.nn.softmax))
```

Texto 12: Código: generación del modelo MLP de 3 capas para usar en la base de datos MNIST.

En este ejemplo, se tienen 3 capas, una de entrada, una oculta y una de salida. La capa de entrada acepta como indicación las dimensiones de los datos que debe procesar, y como puede verse es de tipo 'Dense', que en el argot de Keras se refiere a una capa *fully conected*, con todas las neuronas conectadas a todas las neuronas de la siguiente capa. La capa interior sigue una estructura similar a la de entrada y se apila sin más. Finalmente la capa de salida tiene únicamente 9 neuronas, ya que se espera de ella que de una predicción de cada entrada y esperamos que cada imagen sea clasificada con el número que contiene, como son dígitos del 0 al 9, necesitaremos 10 nodos.

```
Epoch 1/3
60000/60000 [==============================] - 5s 90us/step - loss: 0.2594 - acc: 0.9238
Epoch 2/3
60000/60000 [==============================] - 5s 81us/step - loss: 0.1055 - acc: 0.9673
Epoch 3/3
60000/60000 [==============================] - 5s 89us/step - loss: 0.0729 - acc: 0.9766
<tensorflow.python.keras.callbacks.History at 0x1d609e8ac18>
```

Figura 27: Resultado de la clasificación con el modelo secuencial y la base de datos MNIST

```
10000/10000 [==============================] - 1s 58us/step
0.09367659104913473
0.9717
```

Figura 28: Resultado de la validación del modelo secuencial y la base de datos MNIST

Como puede verse, el entrenamiento además de rápido resulta en un modelo muy preciso, con lo que puede decirse que ha sabido generalizar la detección de los dígitos. Se especifica también algunas características del entrenamiento:

- Para hacer muy sencillo el modelo, se utiliza una entrada de dimensiones de vector. Para esto la imagen se aplana y se convierte a dimensiones *(1,784)*. La primera capa entonces demanda datos de ese formato. Keras provee de herramientas para aplanar las matrices de imágenes, y también puede hacerse con la librería numpy, que dispone de la función *reshape*, útil para conversiones de matrices de todo tipo.

- Como método de optimización se utiliza Adam. Adam es conocido entre la comunidad como un buen sistema de optimización para usar en general. Por eso se utiliza habitualmente por defecto ya que es posible cambiarlo de forma sencilla y da buenos resultados en redes neuronales convolucionales, redes multi-capa, en este caso, la MLP responde rápidamente al entrenamiento con gran precisión.

- Se utiliza como función de pérdidas la entropía cruzada. En las redes neuronales se debe recordar que se trata de minimizar el error, no maximizar la precisión. En este caso, entonces la entropía cruzada es un método de comparación entre lo que la red predice y lo que era la respuesta deseada. Elegir la función de pérdidas correctamente es importante ya que difiere de por ejemplo si se desea clasificar entre dos clases y si se desea usar mas de dos clases.

- La métrica, es decir la medida en distancias entre predicción y resultado deseado, que se desea observar es la precisión.

5.8.2 Base de datos de ISIC 2017. Segmentación

Para afianzar los conocimientos se implementa otro modelo, esta vez destinado a la segmentación y sobre un objetivo similar, las lesiones dermatológicas. Como se verá en los últimos capítulos, éste también resulta útil para probar los modelos en conjuntos de datos de mayor cantidad, con la intención de demostrar la validez de los mismos en problemas con bases de datos mejor nutridas que la disponible para el caso del cáncer de ano.

El caso de la base de datos de ISIC también resulta de gran interés. En este caso, la ISIC, la *International Skin Imaging Collaboration* realiza un esfuerzo por mejorar la diagnosis y la detección del melanoma, con el patrocinio de la ISDIS (*International Society for Digital Imaging of the Skin*). Ponen a disposición su enorme archivo de imágenes de dermoscopias.

En las dermoscopias, como las lesiones se encuentran en la superficie de la piel, su evolución es visible al ojo y es razonable intentar detectarlas tempranamente. Debido a esto, también es razonable tratar de detectarlas automáticamente y para esto, la ISIC celebra retos científicos para quienes deseen crear algoritmos con los que facilitar el análisis a los dermatólogos.

La base de datos dispone de 23906 imágenes de tamaños variables y datos que las acompañan, con una descripción de cada imagen en texto realizada por un médico, una nota sobre la localización de la lesión en el cuerpo, una pequeña descripción de la situación médica del paciente, el diagnóstico y el resultado de las biopsias si se han tomado, además de otros datos. Además, se tienen 13779 imágenes de tipo máscara en las que se tiene

en negro el fondo y en blanco la silueta y el interior de la lesión.

El objetivo del uso de la base de datos es triple: automatizar la segmentación de la lesión frente al fondo, acelerar la extracción de características y realizar de forma precisa una clasificación de la lesión. A modo de experimentación y aprendizaje se realiza la primera fase, la segmentación.

Para realizar una segmentación automatizada con técnicas de *deep learning*, es necesario contar con los ejemplos de entrada, las imágenes originales tomadas por dermatoscopia y con las máscaras, que debe realizarlas un profesional para poder asumir que son la verdad de referencia. De esta forma, se trabaja alimentando la entrada y la salida de la red, forzando a que modifique el valor de los pesos de las capas ocultas hasta que sea capaz de realizar correctamente la segmentación basada en la generalización de los ejemplos vistos.

De entre las variadas arquitecturas disponibles para realizar trabajos de segmentación en imágenes, una de las mas utilizadas y sencillas es la U-Net. En su artículo original, se describe la arquitectura U-Net como una red convolucional para segmentación de imagen biomédica. La estructura de la red U da el nombre a la misma, como puede verificarse viendo la figura 29.

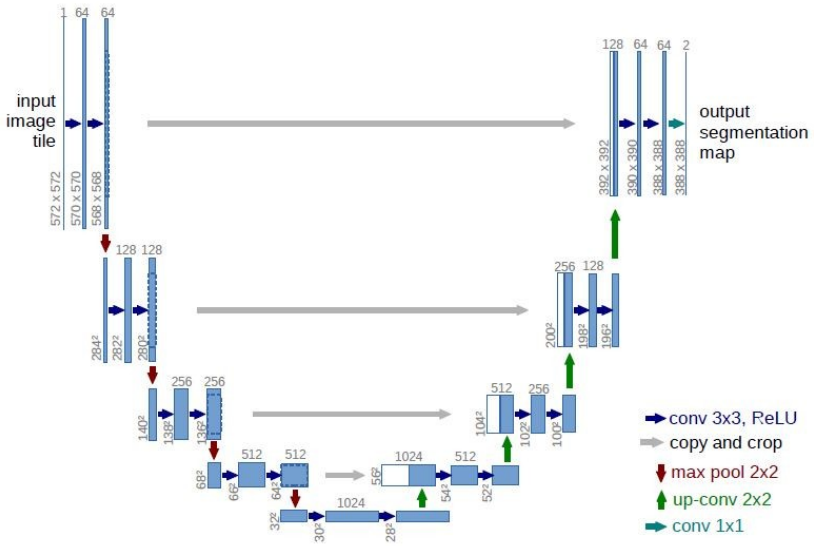


Figura 29: Estructura de la red U-Net. Ronneberger et al. 2015

La estructura consiste en dos caminos, uno que reduce el tamaño de las imágenes y otro que lo aumenta. El camino de la izquierda sigue la estructura típica de una red neuronal convolucional, primero pasa filtros de convolución seguidos de funciones rectificadoras lineales (ReLU) y seguidamente realiza una operación de *pooling* para retener los valores que son considerados de mayor relevancia en la operación. En cada paso del muestreo de entrada se doblan los canales de características. El camino final, de reconstrucción de los datos se necesita recuperar el muestreo realizando operaciones de interpolación, para continuar con una *up-convolution*, que reduce a la mitad los canales en los que fueron separados los datos. Luego se tiene una concatenación de los recortes a los que se llega a la misma altura del camino de entrada y finalmente se pasan filtros de convolución seguidos de rectificadores lineales, como en la entrada.

La estructura de la red tiene un comportamiento que podría resumirse o simplificarse así: al centrar el análisis en áreas cada vez de menor tamaño en la imagen, la red es capaz de contrastar lo importante que resulta un elemento a distintos niveles, es capaz de buscar, ya que en los primeros pasos, filtra la imagen completa, momento en el que tiene información local de una estructura pequeña y general porque debe situarla en un vecindario y al ir realizando operaciones de submuestreo se centra en elementos de menor tamaño, revelando así la importancia de unas u otras características. Sería como acercar la vista para identificar algo, y luego alejarla para situarla y tenerla controlada en su entorno habitual.

Para realizar pruebas con el vasto paquete de datos del ISIC, se realiza una primera división del conjunto y se extrae un primer conjunto de 2600 imágenes de entrenamiento y 490 de validación. Las imágenes se ajustan para que tengan las mismas dimensiones y terminan siendo de 256 por 256 píxeles.

Sin mas preprocesamiento que la normalización que ofrece Keras, las imágenes son entrenadas. Algunas características del entrenamiento son:

- Se usan pesos de una red entrenada previamente con la base de datos ImageNet. Esto es un paso habitual ya que ese pequeño paso de **Transfer Learning** permite que la red parta de una gran cantidad de trabajo realizado. Es habitual hacer esto porque tras procesar muchas imágenes, normalmente si las primeras capas disponen de filtros convolucionales, se acabarán dedicando a tareas similares: la detección de bordes, la agrupación de pequeñas estructuras y objetos y no es hasta que se llega a las capas posteriores que la información se hace lo suficientemente abstracta como para que el problema se particularice notoriamente. Si se entrenase la red "*from scratch*" o de la nada, se necesitarían muchas mas imágenes para llegar a la detección correcta de las lesiones y en el caso de la imagen médica no es habitual disponer de centenas de miles de muestras.

- De nuevo, como criterio de optimización se utiliza Adam

- La función de pérdidas ahora es radicalmente distinta a la seleccionada en el ejemplo de la base de datos MNIST. Ahora se trata de minimizar el índice Jaccard. Éste índice mide la diferencia entre dos conjuntos dividiendo el área de intersección entre el área de unión. Siempre toma valores entre 0 y 1. Es un indicador útil y es fácil de implementar en el caso de máscaras binarias, que son resultado de las predicciones de la red. Puede reducirse a ver qué valores debería tener el resultado obtenido y cuáles no, comprobando y contando el total de píxeles que en el resultado deseado y en la predicción se tienen. Puede verse así que en función de la tarea, en las redes neuronales debe afinarse cada parámetro en función de la tarea. Dados dos conjuntos A y B que en el caso que ocupa este trabajo representan grupos de píxeles de una imagen, el índice de Jaccard se define:

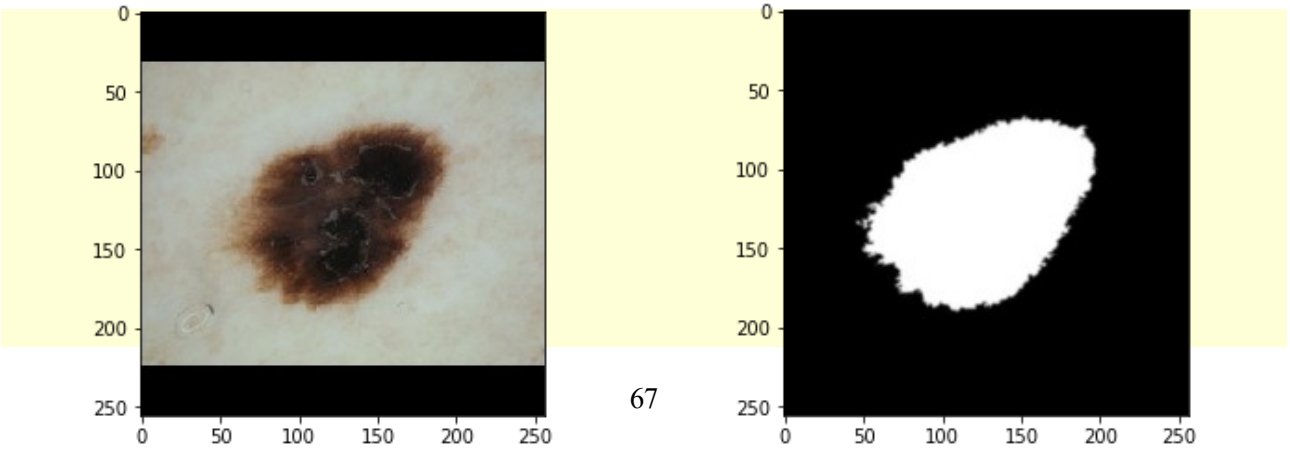
$$\mathrm{J(A,B)} = \frac{AB}{A+B}$$

- Como métrica, se tiene habitualmente en segmentación la puntuación de la IoU o *Intersection of Union*, la intersección de la unión de los conjuntos, es similar al índice Jaccard pero no mide la diferencia, mide el parecido, razón por la cual es usado como métrica.

$$\mathrm{IoU} = \frac{A \cap B}{A \cup B}$$

El modelo de U-Net tiene 23 capas, así que la capacidad es mucho mayor que el caso anterior. La red además, realiza *backpropagation* y tiene capas de filtros convolucionales, así que de nuevo, cada capa realiza operaciones más complejas que el caso anterior. Todo esto debe servir para justificar la enorme diferencia que hay entre entrenar en el caso anterior y en éste. Antes se procesaban las 60000 imágenes en apenas segundos, por la estructura de las capas. En esta ocasión, iterar sobre el conjunto de 2000 imágenes de entrenamiento puede llevar aproximadamente una hora en el equipo descrito al principio del método propuesto.

En este caso se realizan entrenamientos casi sin preprocesamiento, las imágenes están únicamente redimensionadas y normalizadas. Tras 50 *epochs* con un *batch size* de 10 se puede ver la evolución de la red, plasmada en la tabla siguiente.



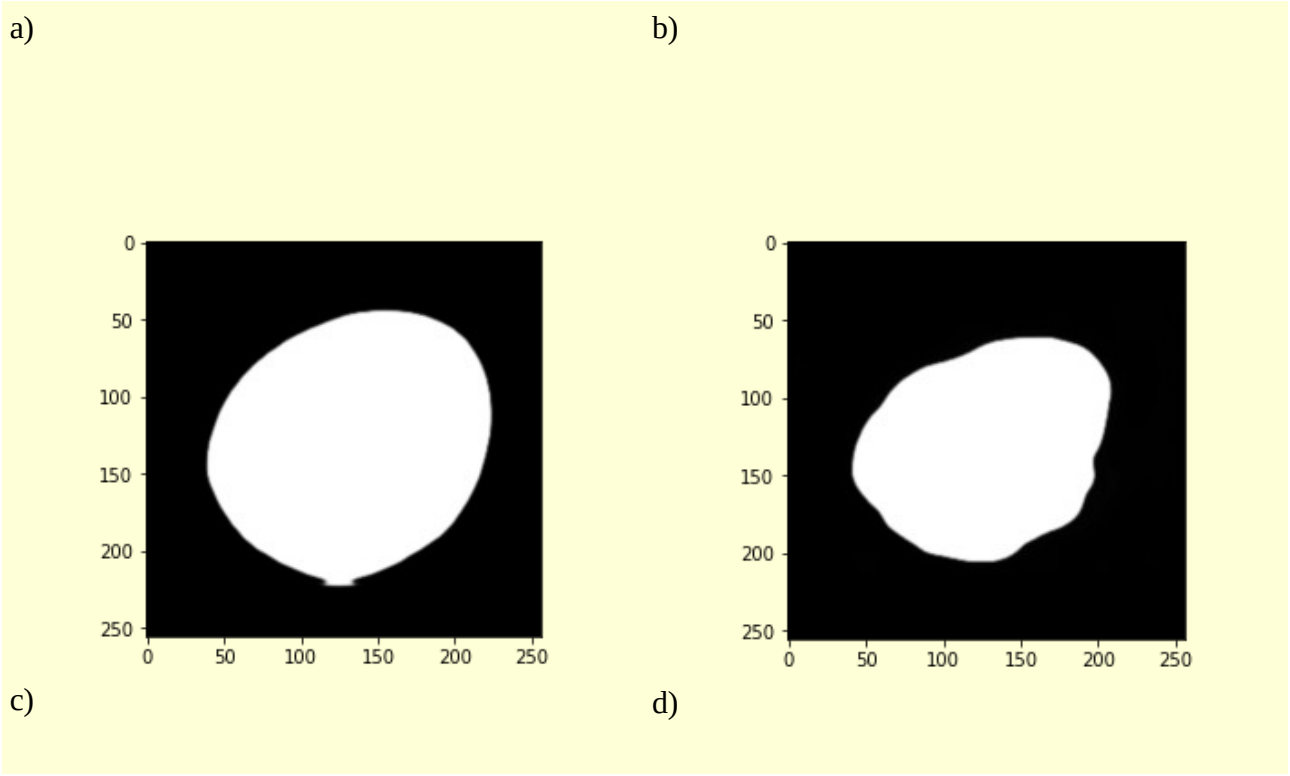


Tabla 7: Muestras de estudio de U-Net. De izquierda a derecha y de arriba a abajo: a) imagen original de dermatoscopia, b) máscara delineada por un profesional, es el resultado deseado, c) máscara predicha por la red tras varios entrenamientos, d) máscara predicha por la red tras entrenar 2 epochs mas que c).

Las pruebas previas son esclarecedoras en algo que se ha ido comentando: la gran capacidad modular de la forma de aplicar las tecnologías de *deep learning* hoy en día permite una flexibilidad enorme, pudiendo resolver problemas muy diversos y con enfoques aparentemente similares que pueden diferir mucho. Es necesario acumular cierta experiencia para lograr llegar a certezas, ¿qué función de pérdidas es mejor?, ¿debería incluso crear una función de pérdidas que se ajuste al problema de forma mas precisa que una de uso general?, esas cuestiones pueden resolverse con la ayuda de la tendencia que puede verse al trabajar con las tecnologías de deep learning en tareas diversas.

5.8.3 Modelos

Un buen método para iniciar la elección de modelos de redes neuronales de *deep learning* es comenzar por lo más básico y sencillo. Como no hay un nutrido estado del arte, no es intuitivo elegir un modelo sin referencias de investigadores que hayan enfrentado el problema. Por suerte existe la referencia de compañeros que han trabajado en el caso del cáncer de cervix y han utilizado habitualmente el perceptrón multicapa o MLP. Éste es objeto de pruebas en este trabajo ya que sería vano no dar una oportunidad a pesar de que resulta un modelo quizá demasiado sencillo y que la detección del cáncer de cérvix es menos complicada que la del cáncer de ano, razón por la cual puede resultar deseable encontrar un modelo con mayor capacidad.

En 2012 Krizhevsky *et al.* consiguen ganar el reto sobre la base de datos de ImageNet con su red "AlexNet" y desde ese momento, se ha aplicado su trabajo en gran cantidad de áreas y ha inspirado muchas otras arquitecturas. Tiempo después, surgen VGGNet y GoogLeNet que encuentran resultados comparables y entre los más precisos en tareas de clasificación con el conjunto de datos de ImageNet en 2014[ver recursis web] y constatando que en poco tiempo, los modelos de redes convolucionales habían mejorado enormemente, siendo ahora competitivos con sistemas generados a mano con muchas horas de profundo diseño ingenieril.

A continuación se discuten los modelos utilizados para realizar pruebas en el presente proyecto.

5.8.3.1 MLP

Un clasificador de capas densas o secuencial, de tipo alimentado hacia adelante (*feed forward*) es utilizado para intentar realizar la tarea de distinguir las lesiones. Es el ejemplo utilizado al inicio de la descripción propuesta. Su arquitectura tiene la forma que queda representada en la figura 6. Ese ejemplo no es estrictamente el que dicta la arquitectura del modelo MLP utilizado en este proyecto, sin embargo, sí que representa eficazmente la estructura de conexiones. Se dispone de un modelo de varias capas con neuronas conectadas. Cada neurona de una capa está conectada a todas las neuronas de la capa siguiente. En la última capa o capa de salida se tiene que para cada clase que se desee clasificar se tendrá una neurona que al activarse con mayor fuerza que las demás, generará una predicción favorable a la clase que corresponda.

Este modelo de red es bastante sencillo o sobrio. Esto incurre en que la detección de patrones resulta compleja y para la clasificación pura podría tener poca capacidad ya que las imágenes de anoscopia resultan bastante difíciles de clasificar.

5.8.3.2 VGG16

La red VGG16 probada, inspirada en la arquitectura VGGNet es una red con un gran número de parámetros y que, inspirada en AlexNet se ha convertido en un clásico entre los modelos de *deep learning* utilizados para la clasificación y la búsqueda de parámetros en imágenes.

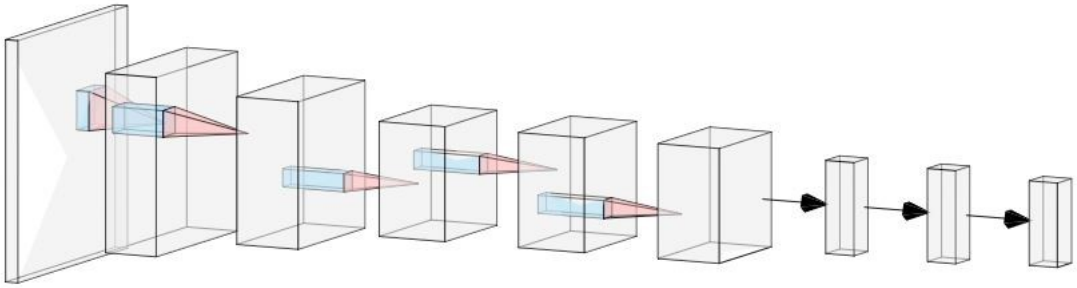


Figura 30: Aspecto de la arquitectura de las redes VggNet

La figura 30 representa el aspecto del modelo. Se ve cómo la red va modificando y transformando los datos, haciendo poco a poco procesado que permite reducir el volumen de datos ya que se está ganando en cada paso un nivel conceptual cada vez mas abstracto. El procesado es como se ha comentado basado en filtros de convolución y el muestreo del resultado de los mismos. Concretamente la red Vgg16 puede implementarse en su forma genérica y adaptada al trabajo con la base de datos ImageNet según se muestra en el siguiente trozo de código:

```
1.  model = Sequential()
2.  model.add(ZeroPadding2D((1,1))
3.  model.add(Convolution2D(64, 3, 3, activation='relu'))
4.  model.add(ZeroPadding2D((1,1)))
5.  model.add(Convolution2D(64, 3, 3, activation='relu'))
6.  model.add(MaxPooling2D((2,2), strides=(2,2)))

7.  model.add(ZeroPadding2D((1,1)))
8.  model.add(Convolution2D(128, 3, 3, activation='relu'))
9.  model.add(ZeroPadding2D((1,1)))
10. model.add(Convolution2D(128, 3, 3, activation='relu'))
11. model.add(MaxPooling2D((2,2), strides=(2,2)))
12.
13. model.add(ZeroPadding2D((1,1)))
14. model.add(Convolution2D(256, 3, 3, activation='relu'))
15. model.add(ZeroPadding2D((1,1)))
16. model.add(Convolution2D(256, 3, 3, activation='relu'))
17. model.add(ZeroPadding2D((1,1)))
18. model.add(Convolution2D(256, 3, 3, activation='relu'))
19. model.add(MaxPooling2D((4,4), strides=(2,2)))
20.
21. model.add(Flatten())
22. model.add(Dense(4096, activation='relu'))
23. model.add(Dropout(0.5))
24. model.add(Dense(4096, activation='relu'))
25. model.add(Dropout(0.5))
26. model.add(Dense(1, activation='softmax'))
```

Texto 13: Implementación de Vgg16 en Keras

El modelo base está adaptado a su uso con ImageNet ya que clasifica para tantas clases como tiene dicha base de datos y además las dimensiones que el modelo espera encontrar en los datos de entrada son las de las imágenes de ImageNet (224x224 píxeles). Este modelo es cargado con los pesos pre-entrenados tras procesar la mencionada base de datos y luego utilizado para intentar clasificar las imágenes del proyecto. Esto no da buenos resultados ya que las capas entrenadas previamente pierden el trabajo realizado porque deben adaptarse a las nuevas imágenes y esto resulta en que el modelo no termina de encontrar una tendencia de mejora.

Para poder aprovechar un modelo que ha recibido un entrenamiento exhaustivo es conveniente cargarlo y utilizarlo como la base de un nuevo modelo adaptado mejor a la necesidad del proyecto. Esto es lo que finalmente se opta por hacer ya que de otra forma se encuentran resultados muy pobres. Trabajando de forma similar a como se describe en la siguiente sección.

5.8.3.3 Inception V3

El modelo de Inception V3, como el de VGG16 está incluido entre los que trae Keras disponibles para su implementación. Es una versión mejorada de GoogLeNet, la cual ya trata de solventar problemas de VGGNet y de AlexNet. Tiene un comportamiento eficiente, incluso con poca cantidad de memoria disponible, debido a una reducción de parámetros notable, 9 veces menos que AlexNet (VGGNet tiene unas 3 veces mas parámetros que AlexNet). A pesar de que la arquitectura de VGGNet es sencilla, su coste computacional resulta muy elevado y por su parte, Inception está diseñada para ser capaz de funcionar hasta en computadoras económicas.

Un problema que tiene este modelo es su complejidad en la arquitectura, que dificulta poder hacer cambios. Parte de esto es debido a que se factorizan convoluciones para reducir el número de conexiones entre nodos de la red. Se tienen en Inception v3 tres módulos que factorizan convoluciones: se sustituyen filtros de 5x5 por dos capas de filtros de dimensiones 3x3, las convoluciones de tamaño 3x3 se convierten en filtros de 3x1 y 1x3 y módulos que aumentan las dimensiones, pasando de un filtro 1x1 a dos filtros 1x3 y 3x1 con herramientas de interpolación.

Una representación de la arquitectura de Inception se muestra a continuación:

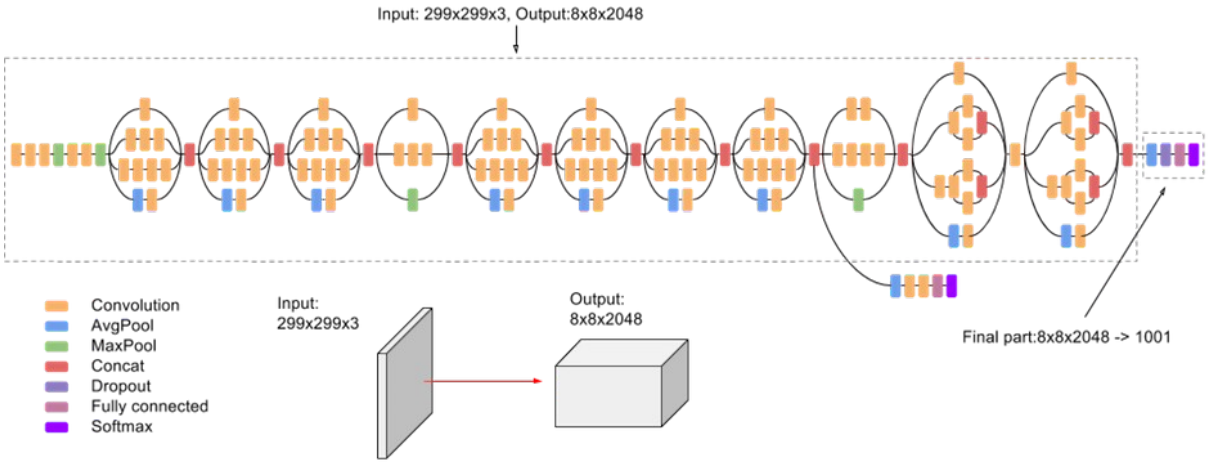


Figura 31: Arquitectura de la red Inception v3. Fuente: https://cloud.google.com/tpu/docs/inception-v3-advanced

Como puede verse, se utilizan los módulos de factorización o reducción de parámetros que resultan en las estructuras en bucle de la figura 31. Con un poco de abstracción en estos elementos, la red puede verse de forma sencilla como una serie de 3 capas con convoluciones seguidas de una de muestreo o *pooling*. Detrás se concatenan otras 3 capas con filtros convolucionales para conectar con: 3 bloques de factorización de 5x5 a dos 3x3, luego 5 bloques que cambian una convolución *n*x*n* por una 1x*n* seguida de una *n*x1. Además se añaden dos bloques de interpolación que aumentan las dimensiones de los datos para acabar haciendo de nuevo un muestreo o *pooling* y finalizar con una capa densa y una función de activación para elegir la clase final.

En el entrenamiento con la red Inception v3 se utiliza el modelo que Keras incluye entre sus módulos de aplicaciones. Así, se puede cargar el modelo base para comenzar a trabajar, como se ve en el código siguiente:

```
27. input_tensor = Input(shape=(1000,1000,3))
28. base_model = InceptionV3(weights='imagenet',include_top=False,input_tensor=input_tensor)
29. x = base_model.output
30. x = GlobalAveragePooling2D()(x)
31. x = Dense(1024, activation='relu')(x)
32. predicciones = Dense(1,activation='sigmoid')(x)
33. model = Model(inputs=base_model.input, outputs = predicciones)
34. for layer in base_model.layers:
35.   layer.trainable= False
36. model.compile(optimizer='adam',loss='binary_crossentropy')
```

Texto 14: Código que carga el modelo base de Inception v3 y lo adapta a la necesidad del proyecto.

En el texto 14 se puede ver en la línea 6 la carga del modelo base. En la línea 8 se añade un *pooling* o muestreo porque la salida de Inception v3 está diseñada para clasificar la base de datos ImageNet que tiene miles de clases, y para poder utilizar el modelo completo, es necesario adaptarlo a nuestro caso, de dos clases. Tras el pooling, se añaden dos capas más, una capa densa que conecta todas las salidas del modelo base con las del modelo adaptado, que en su última capa tiene una capa densa con dos niveles de activación, uno para cada clase y una función sigmoide de activación.

En el caso de un clasificador binario es habitual utilizar una última función de activación en la capa final de tipo sigmoide. Si se entrena un clasificador de más categorías, se suele utilizar una función de activación de tipo softmax.

Finalmente, se toma la precaución de no permitir que las capas del modelo base modifiquen los valores de sus pesos durante el entrenamiento para así, tratar de no perder la capacidad que ésta brinda. Esto se realiza en las líneas 12-13.

```
1.  aug = ImageDataGenerator(rotation_range=20,rescale=1./255,
2.                           horizontal_flip=True,vertical_flip=True)
3.
4.  train_generator = aug.flow_from_directory(
5.                       '/content/drive/My Drive/datos_entrenamiento/train',
6.                       target_size=(1000,1000),
7.                       batch_size=32,
8.                       class_mode='binary')
9.  validation_generator = aug.flow_from_directory(
10.                      '/content/drive/My Drive/datos_entrenamiento/test',
11.                      target_size=(1000,1000),
12.                      batch_size=32,
13.                      class_mode='binary')
```

Texto 15: Código: carga de los datos con flow_from_directory

Sobre este párrafo se muestra la carga de datos, con la capacidad de usar un generador de imágenes para hacer *data augmentation* y evitar el peso en memoria de todas las variantes de cada imagen. En las líneas 1-2 del texto 15 se tiene la declaración del generador: realiza volteos verticales, horizontales, normalización y rotaciones. Al cargar cada imagen, se realizará una de las opciones aleatoriamente durante el entrenamiento.

```
1.  from keras.optimizers import SGD
2.  model.compile(optimizer=SGD(lr=0.000001,momentum=0.9), loss='binary_crossentropy', metrics =
['acc','mse'])
3.  H1 = model.fit_generator( train_generator,
4.                       steps_per_epoch=49,shuffle=True,
5.                       epochs=8,
6.                       validation_data=validation_generator,
7.                       validation_steps=4, verbose=True )
```

Texto 16: Código: entrenamiento de la red Inception v3

En el texto 16 se puede ver cómo se puede controlar el entrenamiento con facilidad: a la hora de compilar el modelo, como en la línea 2 es posible cambiar la función de pérdidas, las métricas, el ratio de aprendizaje, el momento de aprendizaje y muchas otras opciones más. Además puede verse como en la variable H1 de la línea 3 se va a almacenar el historial de la evolución. Otro detalle importante está en la línea 4 y no debe pasar desapercibido, se trata de la directiva shuffle=True, y la idea de incluir esta línea es que los datos estén mezclados aleatoriamente, para que el método *flow_from_directory* no entregue las imágenes de una clase primero, luego la siguiente clase y así. Entregar primero las imágenes de una clase y luego las de la otra clase (en el caso de un clasificador binario) es un error porque cuando se termine el primer conjunto, la red redirigirá la evolución de sus pesos hacia otro destino, la mejor caracterización de la segunda clase, haciendo que la primera parte del entrenamiento no valga para nada.

5.8.4 Conjuntos de imágenes

Para contrastar entre distintos procedimientos, se trabaja con varios conjuntos. Esto permitirá comprobar qué aspectos son relevantes para mejorar la precisión del algoritmo. De esta forma se intenta poder aislar aquellos factores que son relevantes para la mejora del funcionamiento de forma que una vez descubiertos, se pueda hacer

hincapié en ellos.

Debido a la gran cantidad de parámetros que requieren ajuste, se puede trabajar con varios grupos de imágenes pagando el precio de re-entrenar la red. Con estas ideas, se tienen varios conjuntos.

El mas utilizado tiene unas 900 imágenes de recortes y es sometido a pruebas en varios tamaños: cuadrados de 128 por 128 píxeles, de 150 por 150 píxeles, de 224 por 224, de 300 por 300, 500 por 500, 1000 por 1000 y 1500 por 1500.

Estos recortes son preprocesados, normalizados o utilizados en su estado original. Para que, como se ha comentado, se pueda comprender el efecto del uso de una técnica u otra.

Es de ser denotado el hecho de que la búsqueda de patrones en imágenes de enorme tamaño conlleva la caracterización de los parámetros relevantes entre muchas otros datos. Es decir, debe cumplirse un compromiso entre la calidad de las imágenes y el tamaño.

Otra cuestión importante es conseguir que el conjunto, que debe ser dividido en 3 partes: conjunto de entrenamiento, de validación y de test, disponga de muestras en cada grupo que sean i.i.d. Se busca que estén idénticamente distribuidas y que sean independientes. Debido a que en el presente proyecto se cuenta con varias imágenes de los mismos pacientes, la división en conjuntos debe realizarse buscando que un mismo paciente no tenga fotos de la misma sesión en dos grupos diferentes, ya que se estaría jugando contra la independencia de las muestras. Esto se puede hacer de forma sencilla utilizando las referencias de los identificadores de paciente.

En las siguientes tablas se muestran ejemplos de conjuntos de datos de 224 por 224 píxeles. Se utiliza este tamaño porque permite fácil compatibilidad con las redes preentrenadas con ImageNet. Esto se suma a que es un tamaño razonable para ver con definición la enfermedad, en el caso al menos de imágenes recortadas como es el caso que se estudia aquí.

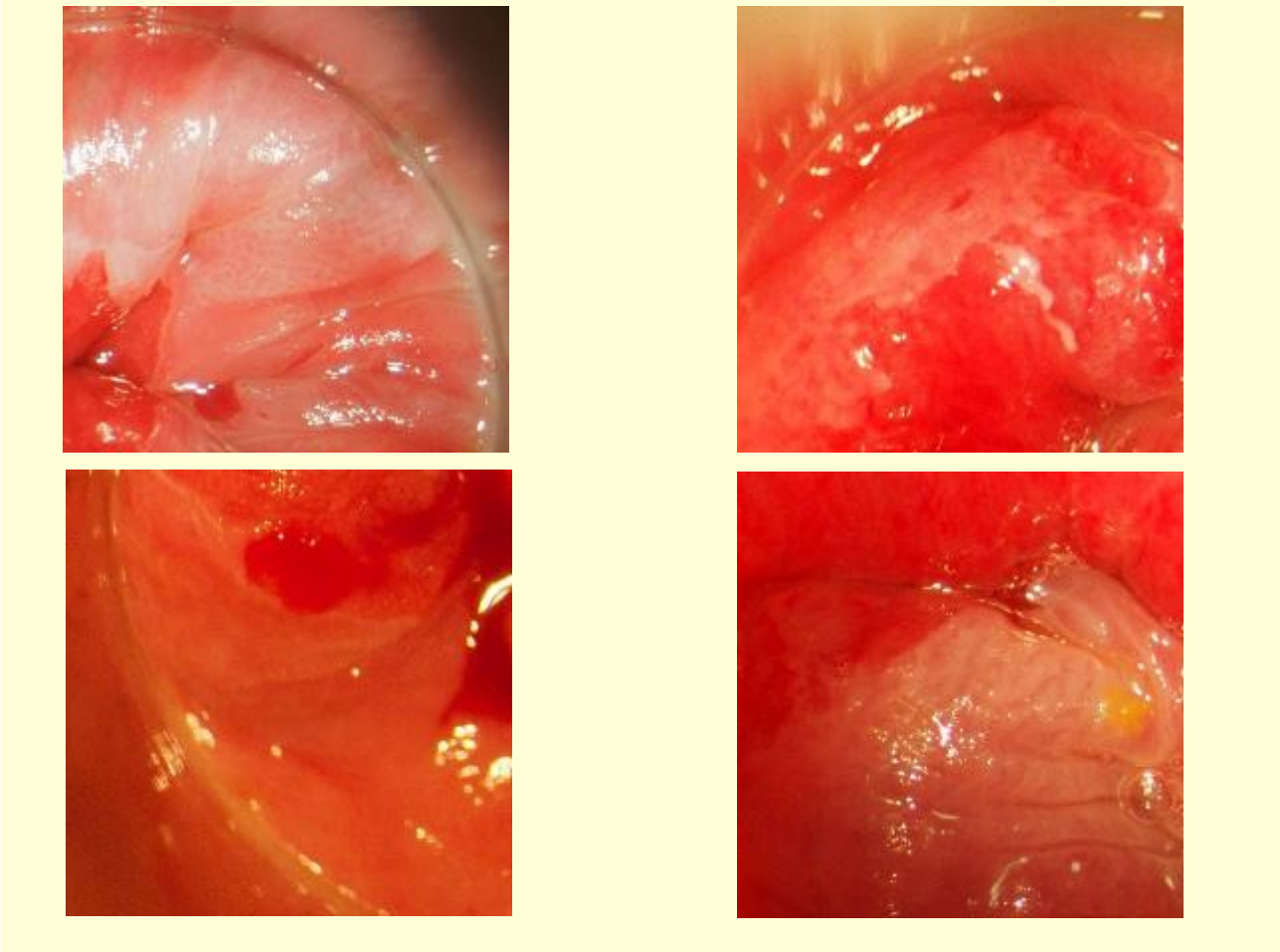


Tabla 8: Ejemplo de imágenes del conjunto usado, imágenes displásicas encima y normales debajo.

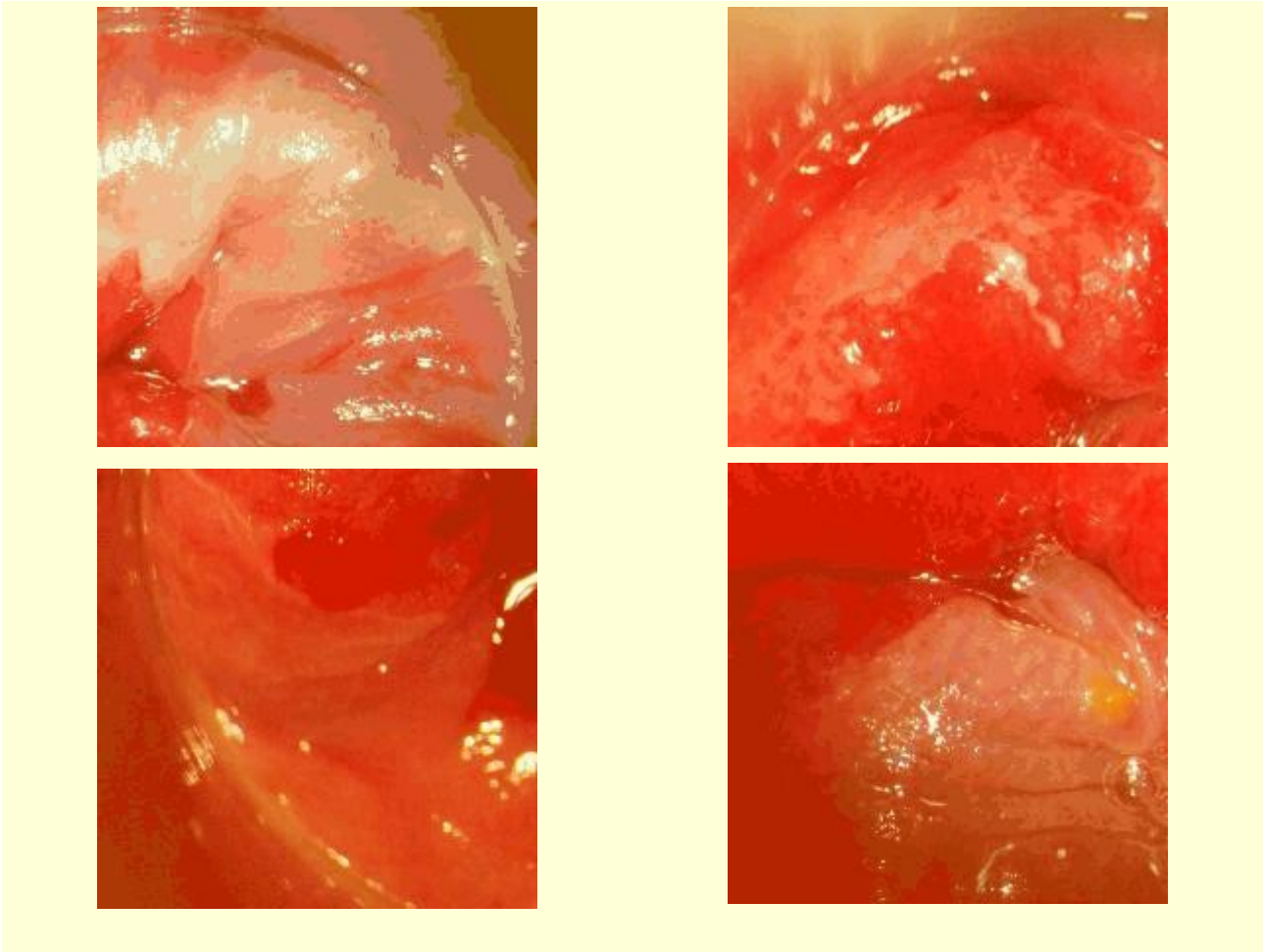


Tabla 9: Ejemplos de imágenes del conjunto preprocesado. Encima: imágenes displásicas, debajo: imágenes normales.

Seguidamente a estos conjuntos de datos se les realiza *data augmentation*. Se generan hasta 5 versiones de cada imagen original al realizar giros y volteos, forzando a la red a descubrir características que son independientes de traslaciones y rotaciones.

6 RESULTADOS

6.1. Resultados de los entrenamientos preliminares en el entorno local

Los resultados de los algoritmos de clasificación y segmentación pueden clasificarse como:

- Verdaderos positivos: si la red predice que existe la enfermedad y se confirma médicamente
- Falsos positivos: si la red predice que existe la enfermedad pero realmente no era así
- Verdaderos negativos: si la red predice que no existe la enfermedad y se confirma médicamente
- Falso negativo: si la red predice que no existe la enfermedad pero realmente no era así

Se pueden generar con estos parámetros datos de mayor interés acerca del progreso del algoritmo. Así puede definirse la sensibilidad, la especificidad, el valor predictivo positivo (PPV), el área bajo la curva (AUC)… Éstos permiten el entendimiento de los resultados así como la comparación entre algoritmos, y dan datos relevantes para el diseño. Estos son valores *a posteriori*, que pueden conseguirse tras realizar pruebas con un algoritmo. Los resultados del algoritmo expuestos arriba son los mismos que se pueden tener tras una prueba diagnóstica en medicina. Con esos datos, es posible comparar el funcionamiento de algoritmos muy dispares o la evolución de uno en concreto a medida que se van realizando modificaciones.

En el caso de un sistema clasificador genérico, un computador podría interpretar los datos como números y los resultados como vectores con probabilidades de que la salida del algoritmo pertenezca a una clase u otra. De esta forma se podría hacer un recuento tras probar a realizar ciertas predicciones para extraer los resultados comentados antes. En el caso de una imagen podemos encontrar varias posibilidades. Si se tiene una tarea de segmentación, por ejemplo, es habitual que se introduzca como dato una imagen y que el resultado también sea una imagen, llamada máscara, en la que se tenga a valor bajo todos los píxeles que no pertenecen a la zona que se deseaba segmentar y a valor alto todos los píxeles que pertenecen a la zona que sí. Si teníamos que la tarea era segmentar una lesión, por ejemplo de piel, la máscara que es resultado del algoritmo puede tener píxeles a valor alto en zonas en las que había lesión (verdaderos positivos), píxeles a valor alto en zonas en las que no había lesión (falsos positivos), píxeles a valor bajo en zonas donde no había lesión (verdadero negativo) y píxeles a valor bajo donde sí había lesión (falso negativo).

Estas pruebas se realizan en el ordenador portátil disponible al inicio del proyecto. En las siguientes secciones se describe el entrenamiento con la red MLP. Debe decirse que también han sido probados los otros modelos en el ordenador portátil, pero como son redes de mucha mayor complejidad, los tiempos de entrenamiento resultan prohibitivos. Debido a esto, se exponen los resultados de la red MLP porque debido a su arquitectura liviana es posible emplearla con entrenamientos mayores que, aun sin dar buenos resultados pueden ser útiles para seguir viendo el proceso de entrenamiento en *deep learning*.

6.1.1 Comentarios respecto al manejo de conjuntos de datos

Las pruebas iniciales se realizan sobre un conjunto de datos reducido. Inicialmente se intenta entrenar la red con unos 300 recortes, algo parecido a lo que disponen algunos de los trabajos relacionados con el proyecto Pompis. Se encuentra una gran dificultad para que con un grupo tan reducido se puedan caracterizar correctamente las cinco clases relevantes en este trabajo. Otra complicación es que las clases con tan pocas muestras tienen a quedar no balanceadas, lo cual provoca que se necesita ajustar de forma fina el peso que cada clase tiene en el

entrenamiento de la red. Esto último se suele solventar de forma simple: se pasa el mismo número exacto de imágenes de cada una de las clases o se controla el peso de cada clase en el entrenamiento siguiendo la función que describe la densidad de probabilidad de las mismas.

Debido a estas cuestiones, los entrenamientos iniciales como se ha comentado no consiguen dar información que permita asumir que se va por un buen camino.

Posteriormente se realiza una división del conjunto de datos en dos clases. Una clase con displasia que incluye los tres grados de AIN, y una clase normal, que incluye las muestras con biopsias normales y los condilomas. Las pruebas en este caso sí parecen dar resultados mejores, de los que se puede partir para sentar las bases de un programa de verdadera ayuda al diagnóstico.

Otra cuestión que se nota relevante en el entrenamiento es la resolución de las imágenes. El uso de recortes de muy pequeño tamaño, como al inicio del proyecto se realiza es un error. En el tamaño elegido, que es de 128 por 128 píxeles, se tiene la idea de que sean cómodos para su procesamiento, ya que cargar en la memoria del computador grupos de datos en cantidades masivas provoca que el trabajo resulte lento. Por eso, en las etapas iniciales en las que hay muchas pruebas, es rentable el uso de los datos en su versión de menor peso. Sin embargo, no parece que los recortes de tan reducido tamaño hayan sido buenos para caracterizar satisfactoriamente la enfermedad. Como se explica en la sección 5.5, se realizan recortes de mayor tamaño sobre las coordenadas de los pequeños y en esta ocasión se dispone de recortes de 1500x1500 píxeles. Esto es útil para poder trabajar con las imágenes en una calidad alta, pudiendo operar con ellas a sabiendas de que no se incurre en pérdidas. A la hora de entrenar las redes sin embargo, se ajustará para encontrar una relación provechosa entre tiempo de entrenamiento/predicción y calidad mínima de las imágenes porque, entrenar las redes con imágenes de tamaños diminutos puede complicar la detección de detalles sutiles y pequeños y hacerlo con las imágenes en tamaños enormes conlleva que se debe buscar la información verdaderamente relevante entre muchos otros datos, lo cual puede generar la necesidad de trabajar con un modelo con mayor capacidad.

6.1.2 Resultados con la red MLP

Las primeras pruebas son realizadas con el modelo mas básico, el MLP de capas densas y tipo secuencial que se puede ver en el texto 12, que tiene todos los nodos de una capa conectados a todos los nodos de la capa siguiente. Se muestran algunos resultados de los entrenamientos:

```
Epoch 1/50
620/620 [==============================] - ETA: 10s - loss: 1.6094 - acc: 0.26 - ETA: 5s - loss: 1.6061 - acc: 0.2500 - ETA: 3s - loss: 1.5901 - acc: 0.276 - ETA: 2s - loss: 1.5707 - acc: 0.289 - ETA: 1s
- loss: 1.5841 - acc: 0.271 - ETA: 1s - loss: 1.5735 - acc: 0.268 - ETA: 0s - loss: 1.5734 - acc: 0.272 - ETA: 0s - loss: 1.5705 - acc: 0.269 - ETA: 0s - loss: 1.5660 - acc: 0.277 - 3s 4ms/sample - loss: 1.5623 - acc: 0.2806
Epoch 2/50
620/620 [==============================] - ETA: 1s - loss: 1.5762 - acc: 0.265 - ETA: 1s - loss: 1.5261 - acc: 0.304 - ETA: 1s - loss: 1.5112 - acc: 0.322 - ETA: 0s - loss: 1.4979 - acc: 0.304 - ETA: 0s
- loss: 1.5237 - acc: 0.300 - ETA: 0s - loss: 1.5103 - acc: 0.307 - ETA: 0s - loss: 1.5203 - acc: 0.301 - ETA: 0s - loss: 1.5243 - acc: 0.296 - ETA: 0s - loss: 1.5279 - acc: 0.289 - 2s 2ms/sample - loss: 1.5299 - acc: 0.2887
Epoch 3/50
620/620 [==============================] - ETA: 1s - loss: 1.5304 - acc: 0.281 - ETA: 1s - loss: 1.5451 - acc: 0.281 - ETA: 0s - loss: 1.5368 - acc: 0.265 - ETA: 0s - loss: 1.5376 - acc: 0.265 - ETA: 0s
- loss: 1.5377 - acc: 0.284 - ETA: 0s - loss: 1.5429 - acc: 0.278 - ETA: 0s - loss: 1.5434 - acc: 0.267 - ETA: 0s - loss: 1.5354 - acc: 0.269 - ETA: 0s - loss: 1.5379 - acc: 0.262 - 2s 2ms/sample - loss: 1.5331 - acc: 0.2629
Epoch 4/50
620/620 [==============================] - ETA: 1s - loss: 1.4914 - acc: 0.375 - ETA: 1s - loss: 1.4843 - acc: 0.351 - ETA: 0s - loss: 1.5538 - acc: 0.270 - ETA: 0s - loss: 1.5363 - acc: 0.300 - ETA: 0s
- loss: 1.5369 - acc: 0.296 - ETA: 0s - loss: 1.5312 - acc: 0.302 - ETA: 0s - loss: 1.5340 - acc: 0.308 - ETA: 0s - loss: 1.5299 - acc: 0.306 - ETA: 0s - loss: 1.5301 - acc: 0.296 - 1s 2ms/sample - loss: 1.5311 - acc: 0.2919
Epoch 5/50
620/620 [==============================] - ETA: 1s - loss: 1.4984 - acc: 0.250 - ETA: 1s - loss: 1.4962 - acc: 0.257 - ETA: 0s - loss: 1.5244 - acc: 0.255 - ETA: 0s - loss: 1.5143 - acc: 0.289 - ETA: 0s
- loss: 1.5063 - acc: 0.303 - ETA: 0s - loss: 1.5097 - acc: 0.291 - ETA: 0s - loss: 1.5208 - acc: 0.296 - ETA: 0s - loss: 1.5221 - acc: 0.287 - ETA: 0s - loss: 1.5206 - acc: 0.291 - 1s 2ms/sample - loss: 1.5202 - acc: 0.2919
Epoch 6/50
620/620 [==============================] - ETA: 1s - loss: 1.6132 - acc: 0.312 - ETA: 1s - loss: 1.5767 - acc: 0.273 - ETA: 0s - loss: 1.5615 - acc: 0.270 - ETA: 0s - loss: 1.5563 - acc: 0.265 - ETA: 0s
- loss: 1.5461 - acc: 0.259 - ETA: 0s - loss: 1.5412 - acc: 0.263 - ETA: 0s - loss: 1.5242 - acc: 0.270 - ETA: 0s - loss: 1.5213 - acc: 0.261 - ETA: 0s - loss: 1.5052 - acc: 0.284 - 2s 3ms/sample - loss: 1.5038 - acc: 0.2919
Epoch 7/50
620/620 [==============================] - ETA: 1s - loss: 1.5155 - acc: 0.343 - ETA: 1s - loss: 1.5187 - acc: 0.304 - ETA: 1s - loss: 1.5188 - acc: 0.291 - ETA: 0s - loss: 1.5298 - acc: 0.300 - ETA: 0s
- loss: 1.5384 - acc: 0.296 - ETA: 0s - loss: 1.5372 - acc: 0.286 - ETA: 0s - loss: 1.5297 - acc: 0.292 - ETA: 0s - loss: 1.5159 - acc: 0.298 - ETA: 0s - loss: 1.5214 - acc: 0.296 - 2s 3ms/sample - loss: 1.5210 - acc: 0.2919
Epoch 8/50
620/620 [==============================] - ETA: 1s - loss: 1.5524 - acc: 0.343 - ETA: 1s - loss: 1.5200 - acc: 0.304 - ETA: 0s - loss: 1.5246 - acc: 0.307 - ETA: 0s - loss: 1.5115 - acc: 0.320 - ETA: 0s
- loss: 1.5138 - acc: 0.309 - ETA: 0s - loss: 1.5215 - acc: 0.296 - ETA: 0s - loss: 1.5101 - acc: 0.310 - ETA: 0s - loss: 1.5125 - acc: 0.310 - ETA: 0s - loss: 1.5120 - acc: 0.305 - 1s 2ms/sample - loss: 1.5088 - acc: 0.3016
Epoch 9/50
620/620 [==============================] - ETA: 1s - loss: 1.6661 - acc: 0.234 - ETA: 1s - loss: 1.5898 - acc: 0.226 - ETA: 0s - loss: 1.5661 - acc: 0.255 - ETA: 0s - loss: 1.5598 - acc: 0.261 - ETA: 0s
- loss: 1.5576 - acc: 0.250 - ETA: 0s - loss: 1.5478 - acc: 0.276 - ETA: 0s - loss: 1.5315 - acc: 0.274 - ETA: 0s - loss: 1.5128 - acc: 0.291 - ETA: 0s - loss: 1.5323 - acc: 0.288 - 1s 2ms/sample - loss: 1.5271 - acc: 0.2903
Epoch 10/50
620/620 [==============================] - ETA: 1s - loss: 1.4458 - acc: 0.328 - ETA: 1s - loss: 1.4557 - acc: 0.320 - ETA: 0s - loss: 1.4858 - acc: 0.291 - ETA: 0s - loss: 1.4812 - acc: 0.281 - ETA: 0s
- loss: 1.4946 - acc: 0.287 - ETA: 0s - loss: 1.5019 - acc: 0.294 - ETA: 0s - loss: 1.5026 - acc: 0.299 - ETA: 0s - loss: 1.4959 - acc: 0.298 - ETA: 0s - loss: 1.5039 - acc: 0.289 - 2s 3ms/sample - loss: 1.5038 - acc: 0.2903
```

Figura 32: Resultado MLP primeras epoch

```
Epoch 40/50
620/620 [==============================] - ETA: 1s - loss: 1.3205 - acc: 0.390 - ETA: 1s - loss: 1.2408 - acc: 0.421 - ETA: 0s - loss: 1.2205 - acc: 0.442 - ETA: 0s - loss: 1.2292 - acc: 0.425 - ETA: 0s
- loss: 1.2596 - acc: 0.403 - ETA: 0s - loss: 1.2450 - acc: 0.395 - ETA: 0s - loss: 1.2415 - acc: 0.401 - ETA: 0s - loss: 1.2408 - acc: 0.396 - ETA: 0s - loss: 1.2658 - acc: 0.388 - 2s 2ms/sample - loss: 1.2599 - acc: 0.3952
Epoch 41/50
620/620 [==============================] - ETA: 1s - loss: 1.2474 - acc: 0.359 - ETA: 1s - loss: 1.1901 - acc: 0.421 - ETA: 1s - loss: 1.2016 - acc: 0.437 - ETA: 0s - loss: 1.2197 - acc: 0.421 - ETA: 0s
- loss: 1.2268 - acc: 0.421 - ETA: 0s - loss: 1.2448 - acc: 0.408 - ETA: 0s - loss: 1.2355 - acc: 0.412 - ETA: 0s - loss: 1.2542 - acc: 0.400 - ETA: 0s - loss: 1.2540 - acc: 0.399 - 2s 3ms/sample - loss: 1.2749 - acc: 0.3871
Epoch 42/50
620/620 [==============================] - ETA: 1s - loss: 1.3822 - acc: 0.406 - ETA: 1s - loss: 1.3331 - acc: 0.390 - ETA: 1s - loss: 1.2839 - acc: 0.411 - ETA: 0s - loss: 1.3371 - acc: 0.394 - ETA: 0s
- loss: 1.3548 - acc: 0.375 - ETA: 0s - loss: 1.3324 - acc: 0.385 - ETA: 0s - loss: 1.3197 - acc: 0.388 - ETA: 0s - loss: 1.3228 - acc: 0.384 - ETA: 0s - loss: 1.3332 - acc: 0.387 - 2s 3ms/sample - loss: 1.3295 - acc: 0.3855
Epoch 43/50
620/620 [==============================] - ETA: 1s - loss: 1.2541 - acc: 0.328 - ETA: 1s - loss: 1.2223 - acc: 0.390 - ETA: 1s - loss: 1.1726 - acc: 0.437 - ETA: 0s - loss: 1.1915 - acc: 0.414 - ETA: 0s
- loss: 1.2312 - acc: 0.403 - ETA: 0s - loss: 1.4199 - acc: 0.380 - ETA: 0s - loss: 1.4545 - acc: 0.368 - ETA: 0s - loss: 1.4325 - acc: 0.380 - ETA: 0s - loss: 1.4455 - acc: 0.361 - 2s 3ms/sample - loss: 1.4498 - acc: 0.3516
Epoch 44/50
620/620 [==============================] - ETA: 1s - loss: 1.4729 - acc: 0.234 - ETA: 1s - loss: 1.4430 - acc: 0.312 - ETA: 1s - loss: 1.4632 - acc: 0.343 - ETA: 0s - loss: 1.4595 - acc: 0.324 - ETA: 0s
- loss: 1.4672 - acc: 0.306 - ETA: 0s - loss: 1.4666 - acc: 0.312 - ETA: 0s - loss: 1.4800 - acc: 0.305 - ETA: 0s - loss: 1.4889 - acc: 0.294 - ETA: 0s - loss: 1.4860 - acc: 0.298 - 2s 3ms/sample - loss: 1.4777 - acc: 0.2903
Epoch 45/50
620/620 [==============================] - ETA: 1s - loss: 1.4519 - acc: 0.343 - ETA: 1s - loss: 1.4577 - acc: 0.281 - ETA: 1s - loss: 1.4782 - acc: 0.286 - ETA: 0s - loss: 1.4598 - acc: 0.296 - ETA: 0s
- loss: 1.4824 - acc: 0.287 - ETA: 0s - loss: 1.4737 - acc: 0.294 - ETA: 0s - loss: 1.4762 - acc: 0.294 - ETA: 0s - loss: 1.4758 - acc: 0.294 - ETA: 0s - loss: 1.4822 - acc: 0.295 - 2s 2ms/sample - loss: 1.4831 - acc: 0.2935
Epoch 46/50
620/620 [==============================] - ETA: 1s - loss: 1.5032 - acc: 0.265 - ETA: 1s - loss: 1.5081 - acc: 0.265 - ETA: 1s - loss: 1.4920 - acc: 0.260 - ETA: 0s - loss: 1.4983 - acc: 0.261 - ETA: 0s
- loss: 1.4808 - acc: 0.293 - ETA: 0s - loss: 1.4710 - acc: 0.299 - ETA: 0s - loss: 1.4735 - acc: 0.294 - ETA: 0s - loss: 1.4784 - acc: 0.285 - ETA: 0s - loss: 1.4751 - acc: 0.288 - 2s 3ms/sample - loss: 1.4711 - acc: 0.2919
Epoch 47/50
620/620 [==============================] - ETA: 1s - loss: 1.5627 - acc: 0.203 - ETA: 1s - loss: 1.4799 - acc: 0.281 - ETA: 1s - loss: 1.4689 - acc: 0.296 - ETA: 1s - loss: 1.4521 - acc: 0.308 - ETA: 0s
- loss: 1.4575 - acc: 0.290 - ETA: 0s - loss: 1.4461 - acc: 0.291 - ETA: 0s - loss: 1.4555 - acc: 0.283 - ETA: 0s - loss: 1.4532 - acc: 0.283 - ETA: 0s - loss: 1.4522 - acc: 0.284 - 2s 3ms/sample - loss: 1.4456 - acc: 0.2968
Epoch 48/50
620/620 [==============================] - ETA: 1s - loss: 1.3700 - acc: 0.328 - ETA: 1s - loss: 1.4088 - acc: 0.296 - ETA: 1s - loss: 1.4145 - acc: 0.307 - ETA: 0s - loss: 1.4450 - acc: 0.277 - ETA: 0s
- loss: 1.4195 - acc: 0.290 - ETA: 0s - loss: 1.4094 - acc: 0.309 - ETA: 0s - loss: 1.3958 - acc: 0.328 - ETA: 0s - loss: 1.3983 - acc: 0.322 - ETA: 0s - loss: 1.3987 - acc: 0.319 - 2s 3ms/sample - loss: 1.3981 - acc: 0.3194
Epoch 49/50
620/620 [==============================] - ETA: 1s - loss: 1.3915 - acc: 0.296 - ETA: 1s - loss: 1.3692 - acc: 0.335 - ETA: 0s - loss: 1.3689 - acc: 0.375 - ETA: 0s - loss: 1.3852 - acc: 0.367 - ETA: 0s
- loss: 1.3777 - acc: 0.371 - ETA: 0s - loss: 1.3784 - acc: 0.367 - ETA: 0s - loss: 1.3621 - acc: 0.372 - ETA: 0s - loss: 1.3410 - acc: 0.375 - ETA: 0s - loss: 1.3539 - acc: 0.373 - 2s 3ms/sample - loss: 1.3486 - acc: 0.3726
Epoch 50/50
620/620 [==============================] - ETA: 1s - loss: 1.3604 - acc: 0.359 - ETA: 1s - loss: 1.2793 - acc: 0.406 - ETA: 1s - loss: 1.2795 - acc: 0.390 - ETA: 0s - loss: 1.2655 - acc: 0.378 - ETA: 0s
- loss: 1.2905 - acc: 0.356 - ETA: 0s - loss: 1.2906 - acc: 0.362 - ETA: 0s - loss: 1.2994 - acc: 0.348 - ETA: 0s - loss: 1.2863 - acc: 0.353 - ETA: 0s - loss: 1.2930 - acc: 0.355 - 2s 3ms/sample - loss: 1.2983 - acc: 0.3548
Out[20]:
<tensorflow.python.keras.callbacks.History at 0x2a810ab0e48>
```

Figura 33: Resultado de entrenamiento con la red MLP ultimas epoch

Como se ve en las figuras sobre este párrafo, la red no es capaz de superar la precisión del 30-35%. Este entrenamiento se realiza con un conjunto de 620 imágenes y las predicciones pueden verse en la siguiente figura:

```
Proporción de aciertos:  0.39285714285714285
Total de aciertos:  22
Total de fallos:  34

3 1 1 1 0 4 1 4 3 1 0 1 0 4 4 1 4 1 1 0 1 1 2 4 0 1 2 1 1 1 0 1 3 2 1 4 0 0 4 0 1 0 2 2 1 1 1 1 0 0 0 2 2 1 2 4 3 3   Clases de entrenamiento (verdades de referencia)
2 0 0 0 0 4 4 2 2 1 2 1 0 2 2 2 4 1 0 0 2 0 2 2 2 0 2 2 2 0 1 2 2 0 4 0 0 0 0 0 1 1 2 0 2 2 0 0 0 1 1 2 4 4 2 0   Clases resultado de la predicción
```

Figura 34: Resultados de predicción del MLP

El resultado es mejor de lo esperado comparado con el resto de intentos. Se ha cambiado el *batch size*, primero se probó con tamaños menores, pero la precisión era mucho peor. El conjunto de imágenes es mayor y se ha intentado tener un balance mejor entre las clases. Parece que la clase 0 (condiloma) no ha sido bien caracterizada, y eso que es visualmente muy sencilla de reconocer, pero esto puede deberse al tamaño de los patrones, que no encierran los condilomas enteros, que pueden a simple vista observarse e identificarse. Entonces, quizá con mas recortes de condilomas, la red realizaría mejor la predicción y generaría menos falsos positivos en la clase condiloma. A pesar de que se ha predecido en exceso la clase condiloma, puede verse que la red ha variado su predicción, lo suficiente como para poder interpretar su búsqueda, ha comenzado de verdad a caracterizar los patrones.

```
Proporción de aciertos:  0.21951219512195122
Total de aciertos:  9
Total de fallos:  32

1 0 3 4 4 0 2 4 0 0 2 1 4 0 0 4 1 1 0 3 2 1 2 3 4 1 2 3 2 4 4 1 0 0 3 1 3 4 2 1 3        Clases de entrenamiento (verdad de referencia)
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1        Clases resultado de la predicción
```

Figura 35: Resultado erróneo de la clasificación con el modelo MLP

En las figuras 34 y 35 pueden verse ejemplos de validaciones con poblaciones pequeñas de datos. Estas predicciones se realizan sobre un conjunto de prueba con clases distribuidas aleatoriamente. En el primer caso puede verse cómo el algoritmo cambia su predicción en cada clase, lo cual indica que no ha sabido caracterizar correctamente las clases y en el segundo siempre mantiene la misma predicción así que el veredicto es el mismo. Sin embargo, en estas pruebas se puede ver una seña del comportamiento de la red.

En el caso de la figura 34 vemos cómo la clase que mejor detecta es la clase 0 (condiloma) y es que es la clase visualmente más sencilla de identificar. Además, la clase 1 (normal) es la clase que más veces el algoritmo

clasifica como perteneciente a la 0. De esta forma es posible que los recortes con displasia sean más semejantes entre sí que los que no tienen displasia (clases 0, condiloma y 1, normal).

En la figura 35 se ve que la red siempre predice los datos como pertenecientes a la clase 1. Esto puede deberse a varias cosas. Un conjunto de entrenamiento que tiene un número diferente de muestras para cada clase puede hacer que la red neuronal se comporte así. En el caso de imágenes con clases que son tan parecidas como en este proyecto, si además no hay balance es posible obtener estos resultados. Además hay otras posibilidades, por ejemplo si se eligen funciones de activación en la última capa erróneas.

En la salida de un clasificador de red neuronal se tiene un vector de salida sin procesar, por ejemplo *[0.2,-1.5,-0.3,2.6]*. Si se clasifica en grados de lesión, sería deseable llegar a un resultado similar a *"probabilidad de un 53% de tener lesión de alto grado"*. Para convertir esos resultados en probabilidades se utiliza una función típicamente sigmoide o softmax. Estas funciones dan probabilidades diferentes para los mismos valores. Por un lado la función sigmoide asume que las probabilidades son independientes, y trata los valores por separado. La salida de una función softmax es un conjunto de probabilidades que en total suman 1, así que se asume que no son independientes.

En el caso de la detección de enfermedades es habitual utilizar la función sigmoide debido a que los casos son no exclusivos, puede haber más de una respuesta correcta. El uso de una función errónea puede, como se ha explicado desencadenar un comportamiento no esperado como el mostrado antes.

6.2. Resultados de entrenamientos en la nube: Google Colaboratory

6.2.1 Resultados con Vgg16

Los resultados de la red Vgg16 manteniendo del modelo todas las capas con la capacidad de actualizar sus pesos resulta infructuoso. De hecho, inicialmente se carga un modelo idéntico al original, pero que en lugar de cargar por defecto imágenes de 224x224 (ajustadas para trabajar con ImageNet) y tener en la última capa una activación de 1000 nodos para las clases de la mencionada base de datos tiene una entrada ajustada a las imágenes de prueba que se utilizan (entre los 300x300 y 1000x1000 son las resoluciones probadas, tratando de mantener la mayor resolución posible que no agote los recursos del ordenador) y una última capa modificada para entrenar con 2 ó 5 clases. Estos entrenamientos no dan buenos resultados, es preferible utilizar un modelo más simple con el que se tenga confianza en no perder el entrenamiento previo (para el cual se descargan los pesos pre-entrenados que la red alcanza tras procesar exhaustivamente una base de datos) o agregar a un modelo pre-entrenado un módulo que se adapte a las necesidades y permita utilizar sin re-entrenar todo el potencial del modelo elegido.

Para trabajar con la red Vgg16 es necesario adaptar el tamaño de los recortes. A la hora de cargar los conjuntos de imágenes se utiliza una cantidad de memoria y los modelos de reconocimiento de patrones también necesitan bastante, por el uso de muchos parámetros a la vez así que se utilizan recortes de 800x800 píxeles, tamaño que

```
Epoch 1/4
49/49 [==============================] - 190s 4s/step - loss: 0.6598 - acc: 0.5927 - mean_squared_error: 0.2338 - val_loss: 0.8084 - val_acc: 0.2544 - val_mean_squared_error: 0.3062
Epoch 2/4
49/49 [==============================] - 182s 4s/step - loss: 0.6564 - acc: 0.6155 - mean_squared_error: 0.2320 - val_loss: 0.8000 - val_acc: 0.2368 - val_mean_squared_error: 0.3022
Epoch 3/4
49/49 [==============================] - 183s 4s/step - loss: 0.6492 - acc: 0.6256 - mean_squared_error: 0.2285 - val_loss: 0.8094 - val_acc: 0.2400 - val_mean_squared_error: 0.3067
Epoch 4/4
49/49 [==============================] - 181s 4s/step - loss: 0.6563 - acc: 0.6187 - mean_squared_error: 0.2320 - val_loss: 0.7852 - val_acc: 0.2895 - val_mean_squared_error: 0.2950

Epoch 1/6
49/49 [==============================] - 184s 4s/step - loss: 0.6498 - acc: 0.6230 - mean_squared_error: 0.2288 - val_loss: 0.7955 - val_acc: 0.2719 - val_mean_squared_error: 0.3001
Epoch 2/6
49/49 [==============================] - 181s 4s/step - loss: 0.6497 - acc: 0.6422 - mean_squared_error: 0.2287 - val_loss: 0.7863 - val_acc: 0.2719 - val_mean_squared_error: 0.2955
Epoch 3/6
49/49 [==============================] - 179s 4s/step - loss: 0.6520 - acc: 0.6253 - mean_squared_error: 0.2298 - val_loss: 0.7720 - val_acc: 0.3000 - val_mean_squared_error: 0.2886
Epoch 4/6
49/49 [==============================] - 176s 4s/step - loss: 0.6468 - acc: 0.6592 - mean_squared_error: 0.2272 - val_loss: 0.7841 - val_acc: 0.2895 - val_mean_squared_error: 0.2945
Epoch 5/6
49/49 [==============================] - 176s 4s/step - loss: 0.6486 - acc: 0.6492 - mean_squared_error: 0.2281 - val_loss: 0.7601 - val_acc: 0.3333 - val_mean_squared_error: 0.2828
Epoch 6/6
49/49 [==============================] - 178s 4s/step - loss: 0.6477 - acc: 0.6494 - mean_squared_error: 0.2277 - val_loss: 0.7577 - val_acc: 0.3100 - val_mean_squared_error: 0.2817

Epoch 1/4
49/49 [==============================] - 183s 4s/step - loss: 0.6491 - acc: 0.6404 - mean_squared_error: 0.2283 - val_loss: 0.7715 - val_acc: 0.3246 - val_mean_squared_error: 0.2884
Epoch 2/4
49/49 [==============================] - 180s 4s/step - loss: 0.6513 - acc: 0.6418 - mean_squared_error: 0.2294 - val_loss: 0.7643 - val_acc: 0.3158 - val_mean_squared_error: 0.2849
Epoch 3/4
49/49 [==============================] - 181s 4s/step - loss: 0.6432 - acc: 0.6531 - mean_squared_error: 0.2255 - val_loss: 0.7490 - val_acc: 0.3800 - val_mean_squared_error: 0.2774
Epoch 4/4
49/49 [==============================] - 175s 4s/step - loss: 0.6480 - acc: 0.6410 - mean_squared_error: 0.2278 - val_loss: 0.7604 - val_acc: 0.3509 - val_mean_squared_error: 0.2829
```

Figura 36: Resultados del entrenamiento de la red Vgg16

resulta suficiente para comparar con los utilizados de 1000x1000.

En la figura 36 se muestran cuatro entrenamientos realizados sobre el mismo modelo. Se va haciendo en fases porque el equipo disponible es portátil y no puede mantenerse fijo realizando entrenamientos muy largos, pero el estado del modelo y sus parámetros pueden guardarse para continuar trabajando en otro momento.

Puede verse que el resultado alcanza un 65% de precisión. La precisión de la validación es sin embargo inferior, porque es un conjunto de validación sin balance y la población de muestras es mucho menor que en el caso de entrenamiento.

Puede notarse también que los tiempos de ejecución de cada *epoch* son de algo mas de 3 minutos, que no es tan poco tiempo si se piensa la pequeña cantidad de imágenes con las que está trabajando el algoritmo. Aún así en esas 14 *epoch* puede verse que las pérdidas en entrenamiento y validación bajan y la precisión en entrenamiento y validación aumentan.

Con tan pocas *epoch*, es complicado aceptar como bueno el resultado, ya que como se verá en siguientes secciones, entrenamientos más largos y con más muestras de datos pueden hacer que la red entre en sobreajuste. La razón por la cual aquí resulta difícil ver ese sobreajuste es porque con imágenes de tamaños tan grandes le resulta a la red mucho más complicado memorizar al detalle todas las imágenes pero, es probable que esperando el tiempo y las *epoch* necesarias (quizá con 25-50) se vieran indicios de ese sobreajuste.

6.2.2 Resultados con el modelo Inception v3

El entrenamiento de la red Inception v3 aparentemente parece bueno. Al procesar el conjunto de recortes de 900 unidades sin apenas preprocesamiento alguno es capaz de alcanzar un 60% de precisión en la clasificación. Este resultado no es lo suficientemente bueno como para sustituir a un médico, pero la tendencia al alza indica que es viable utilizar estas técnicas para intentar hacer la detección de las lesiones displásicas precancerígenas del cáncer ano-rectal.

```
Epoch 1/5
49/49 [==============================] - 410s 8s/step - loss: 0.6608 - acc: 0.5767 - mean_squared_error: 0.2341 - val_loss: 0.6816 - val_acc: 0.5614 - val_mean_squared_error: 0.2432
Epoch 2/5
49/49 [==============================] - 419s 9s/step - loss: 0.6536 - acc: 0.5780 - mean_squared_error: 0.2326 - val_loss: 0.6769 - val_acc: 0.5789 - val_mean_squared_error: 0.2416
Epoch 3/5
49/49 [==============================] - 407s 8s/step - loss: 0.6570 - acc: 0.5794 - mean_squared_error: 0.2342 - val_loss: 0.6772 - val_acc: 0.5614 - val_mean_squared_error: 0.2427
Epoch 4/5
49/49 [==============================] - 405s 8s/step - loss: 0.6562 - acc: 0.5847 - mean_squared_error: 0.2326 - val_loss: 0.6755 - val_acc: 0.6053 - val_mean_squared_error: 0.2406
Epoch 5/5
49/49 [==============================] - 410s 8s/step - loss: 0.6555 - acc: 0.5860 - mean_squared_error: 0.2322 - val_loss: 0.6790 - val_acc: 0.5175 - val_mean_squared_error: 0.2424

Epoch 1/4
49/49 [==============================] - 466s 10s/step - loss: 0.6571 - acc: 0.5919 - mean_squared_error: 0.2311 - val_loss: 0.6686 - val_acc: 0.5877 - val_mean_squared_error: 0.2367
Epoch 2/4
49/49 [==============================] - 466s 10s/step - loss: 0.6498 - acc: 0.6026 - mean_squared_error: 0.2300 - val_loss: 0.6637 - val_acc: 0.6404 - val_mean_squared_error: 0.2364
Epoch 3/4
49/49 [==============================] - 453s 9s/step - loss: 0.6535 - acc: 0.5970 - mean_squared_error: 0.2316 - val_loss: 0.6614 - val_acc: 0.6930 - val_mean_squared_error: 0.2327
Epoch 4/4
49/49 [==============================] - 454s 9s/step - loss: 0.6512 - acc: 0.6012 - mean_squared_error: 0.2296 - val_loss: 0.6717 - val_acc: 0.6404 - val_mean_squared_error: 0.2398
```

Figura 37: Resultado de dos entrenamientos en la red Inception v3

Esta mejora de la precisión indica que durante el entrenamiento la red está siendo capaz de identificar aspectos que ayudan a denotar las clases. La tendencia es prometedora ya que:

- Las imágenes tienen clases que resultan muy parecidas y aún así es capaz de mejorar su comportamiento.
- Para el reducido número de imágenes recortadas con las que entrenar se tiene una dirección de mejora, así que parece que no es descabellado pensar que agregando más recortes al conjunto de entrenamiento la precisión será mucho mejor.
- El entrenamiento encuentra gran dificultad para superar el porcentaje de precisión del 60%. Esto hace pensar que con los escasos recortes usados para entrenar y la falta de un preprocesamiento más agresivo, no es posible alcanzar una mejoría notable.

Tras las pruebas anteriores se realizan otras con un pequeño incremento en el procesado de las imágenes, tratando de buscar la mencionada mejora. Con la herramientas de Keras se hace una transformación de los valores de los píxeles para que se estandaricen condicionados a las características del conjunto de datos. Para esto, se debe hacer un procesado estadístico, ya que se hace sobre todo el conjunto de datos para encontrar los valores medios del conjunto.

Estas pruebas nuevas no parecen aumentar el umbral que antes se menciona de ese 60%. Pero si es cierto que llega a ese punto de convergencia de forma mas rápida, en pocas *epochs*, o iteraciones sobre el total de los datos.

6.3. Resultados del entrenamiento en una máquina local de gran potencia

Es en este punto donde se realizan las pruebas de mayor duración en tiempo de entrenamiento y mayor número de *epochs*.

6.3.1 Entrenamiento de la red Vgg16

La red Vgg16 tiene gran capacidad y en entrenamientos como el que se plantea puede llegar a tener incluso demasiada. Esto puede verse en los resultados de entrenamiento de las figuras mostradas a continuación.

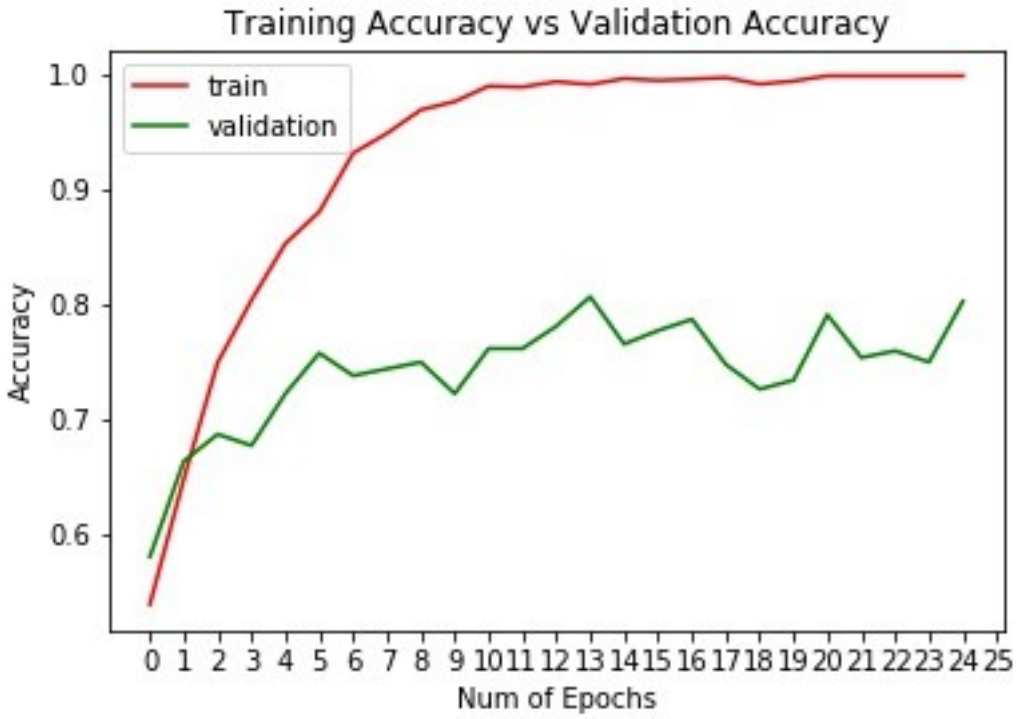


Figura 38: Resultado del entrenamiento de Vgg16. En rojo la precisión de entrenamiento y en verde la precisión de validación.

En la figura 38 puede observarse el resultado del entrenamiento en 25 *epoch*. Se utiliza un tamaño de lote o *batch size* de 32. Los resultados muestran un claro sobreajuste. En tantas vueltas al conjunto de entrenamiento, la red es capaz de memorizar tales muestras y consigue alcanzar una precisión del 100%. La forma que se tiene de confirmar que la situación es tal es precisamente con el conjunto de validación. Si realmente la red hubiera generalizado por completo la solución, la capacidad de la misma para predecir correctamente en imágenes nuevas sería mayor. Como la precisión en el conjunto de validación se estanca en el 0.7, se puede confirmar que el modelo ha entrado en sobreajuste.

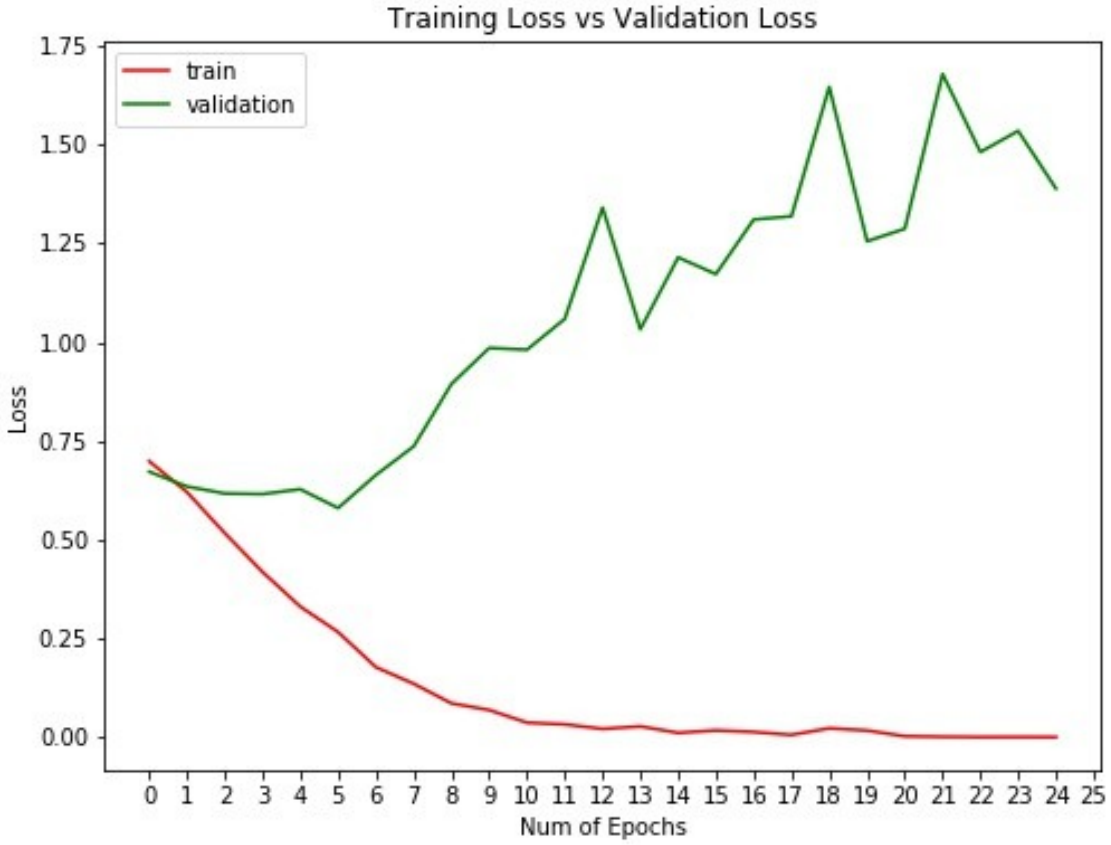


Figura 39: Resultado de la red Vgg16. Pérdidas entrenando en rojo y pérdidas en el conjunto de validación en verde.

En cuanto a las pérdidas, la figura 39 muestra el resultado habitual o lógico en rojo durante el entrenamiento, y el resultado en la validación es sin embargo mucho peor. Esto confirma lo expuesto antes: la red está cada vez menos segura de que durante la predicción con el conjunto de validación las imágenes pertenezcan a una clase u otra. Aunque en la figura 38 la precisión de validación tiene un ligero crecimiento, las pérdidas crecen, esto puede ser debido a situaciones de mayor incertidumbre. Por ejemplo, puede haber muchas fotos con zonas de displasia en las que la red dé como resultado: 20% de probabilidad de ser imagen normal y 80% de probabilidad de estar frente a una lesión displásica pero si durante el entrenamiento se pasa de esa situación a la siguiente: 40% de probabilidad de ser una imagen normal y 60% de probabilidad de ser una imagen displásica, se tendrá la situación mostrada. De esta forma, la red mejora (aunque muy lentamente) su precisión pero aumenta sus pérdidas.

Estos resultados confirman un hecho que es la mayor dificultad a la que se enfrenta el problema: la falta de imágenes de prueba limita los resultados enormemente. Se prueba un modelo InceptionV3 que ofrece resultados enormemente similares.

6.3.2 Entrenamiento de una red convolucional sencilla: ConvNet

El otro modelo probado en una máquina tan potente es una red convolucional sencilla. Puede verse el modelo en el siguiente bloque de código, del que pueden verse todas las capas y operaciones desarrolladas durante el trabajo. El uso de este modelo se realiza porque una menor capacidad podría tener más dificultad para memorizar los casos y ser capaz de resolver el problema con una precisión aceptable.

```
model = Sequential()
model.add(ZeroPadding2D((1,1),input_shape=input_shape))
model.add(Conv2D(64, (11,11)))
model.add(ZeroPadding2D((1,1)))
model.add(Activation('relu'))
model.add(MaxPooling2D((2,2),strides=(2,2)))

model.add(Conv2D(64, (9,9)))
model.add(ZeroPadding2D((1,1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
model.add(Dropout(0.3))

model.add(Conv2D(128, (3,3)))
model.add(ZeroPadding2D((1,1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides=(2,2)))
model.add(Dropout(0.3))

model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

Texto 17: Código: script en el que se genera el modelo de la red convolucional sencilla.

Como puede verse, comparada con la red Vgg16 o la InceptionV3 tiene muchos menos elementos y por ello menor capacidad.

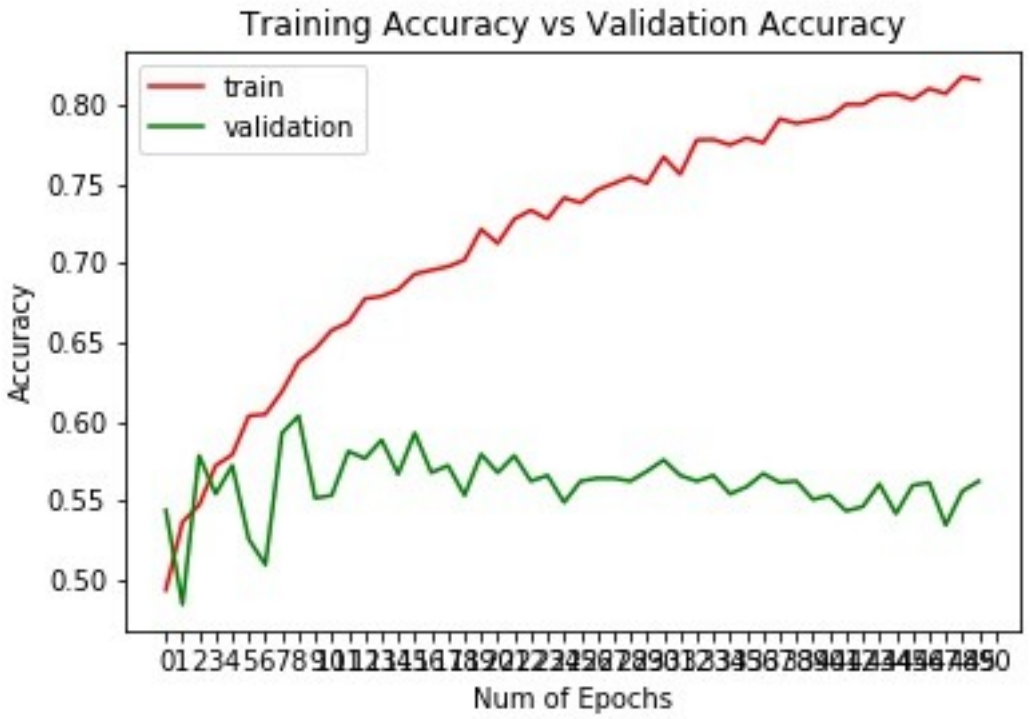


Figura 40: Resultado de la ConvNet. Rojo: precisión en el entrenamiento, verde: precisión de validación.

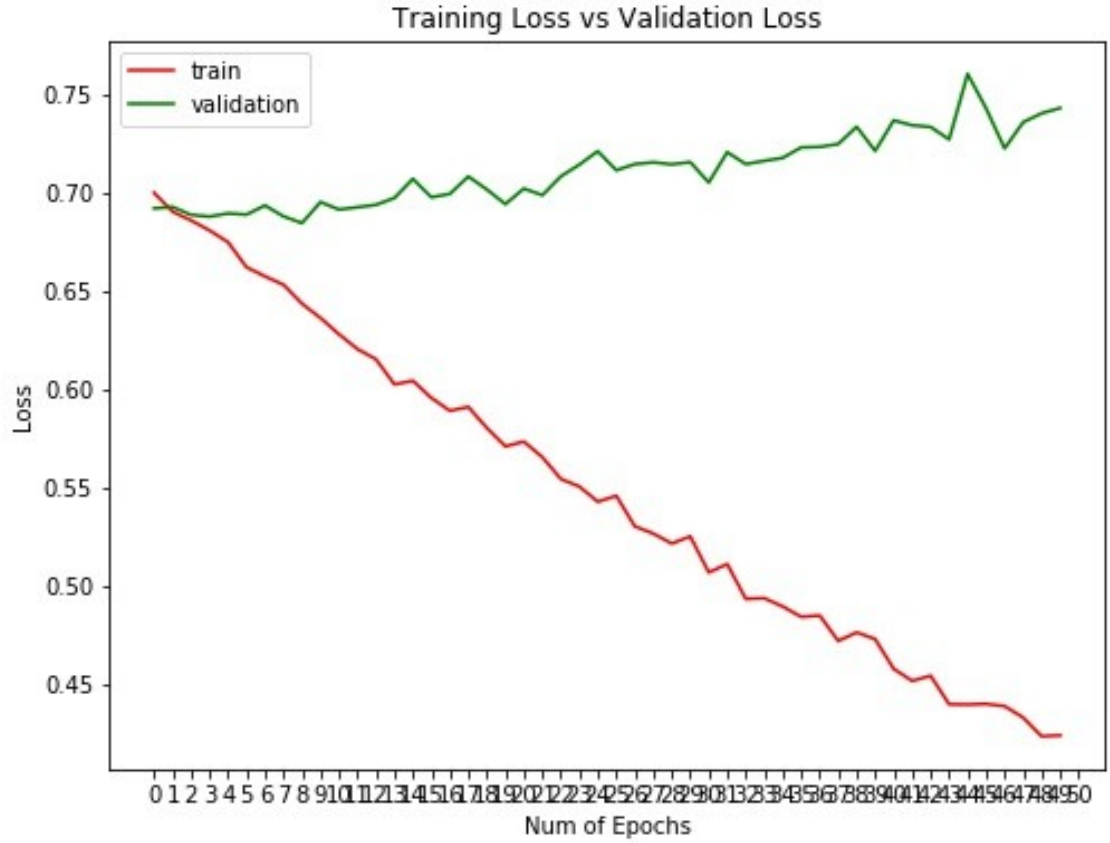


Figura 41: Resultado de la ConvNet. Rojo: pérdidas en el entrenamiento, verde: pérdidas de validación.

Puede verse un comportamiento que recuerda a lo visto en los resultados de la red Vgg16, aunque con comportamientos más controlados, resultados más suaves y con menos picos. En este caso, las pérdidas en la validación no se disparan tanto, y la red no llega a aprender de memoria el 100% de los casos de entrenamiento, aunque esa menor capacidad se nota también en el menor índice de precisión obtenido en la validación.

Si se toma un momento para observar los resultados en la matriz de confusión, puede verse que los resultados no son lo suficientemente buenos como para utilizar el modelo con ese entrenamiento en una solución real, pero hay ciertos detalles que sí pueden ser satisfactorios.

En la figura 42 se tienen cuatro grupos. En el eje vertical se han dispuesto las verdades de referencia: imágenes con displasia e imágenes normales. En el eje horizontal se pueden ver las predicciones, que tienen las mismas clases. Como era una clasificación binaria, la matriz de confusión tiene cuatro grupos. Observando la zona superior, se puede ver que la red tiene dificultades para diferenciar los casos de displasia, ya que se tienen casi los mismos ejemplos clasificados como displasia y como normales. Esto hace que pudieran existir casos en los que el modelo diagnostica como normal sin que así sea, lo cual no es bueno.

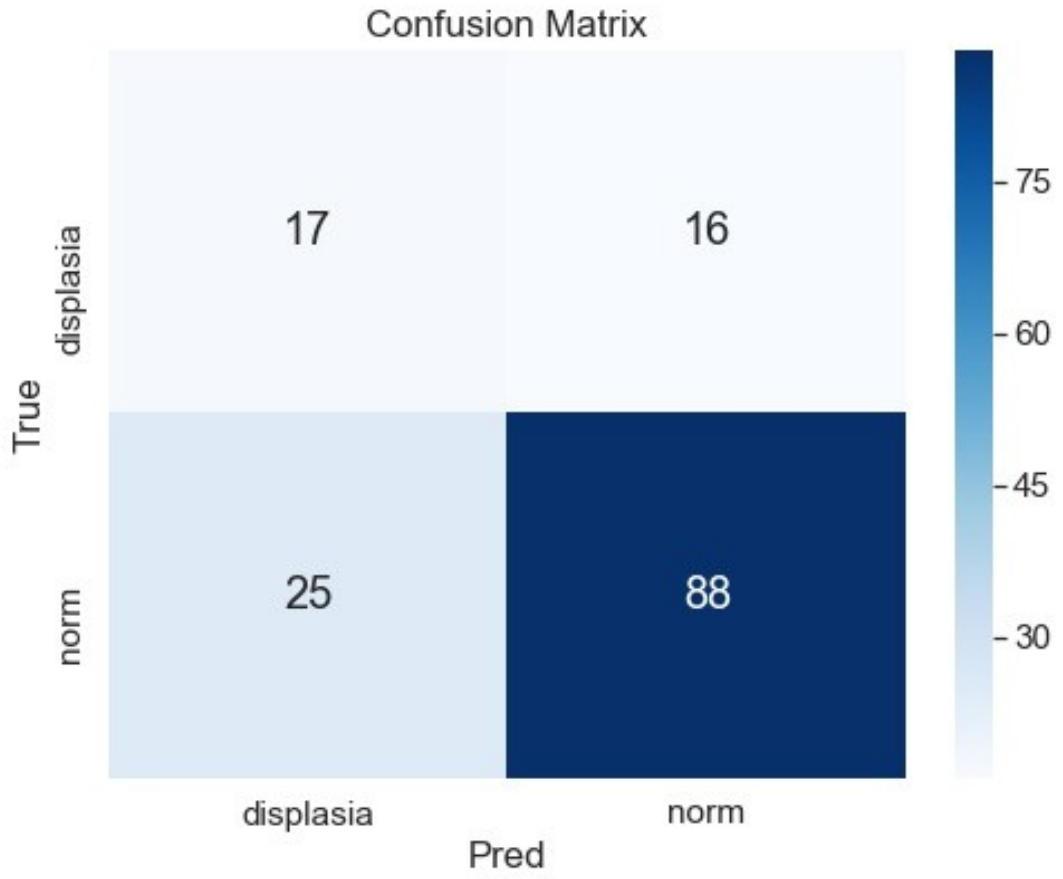


Figura 42: Matriz de confusión resultado del procesado del conjunto de test con la red ConvNet.

Las últimas figuras muestran los resultados de la red ConvNet e imágenes preprocesadas, que presentan resultados similares a los anteriores con algunas diferencias.

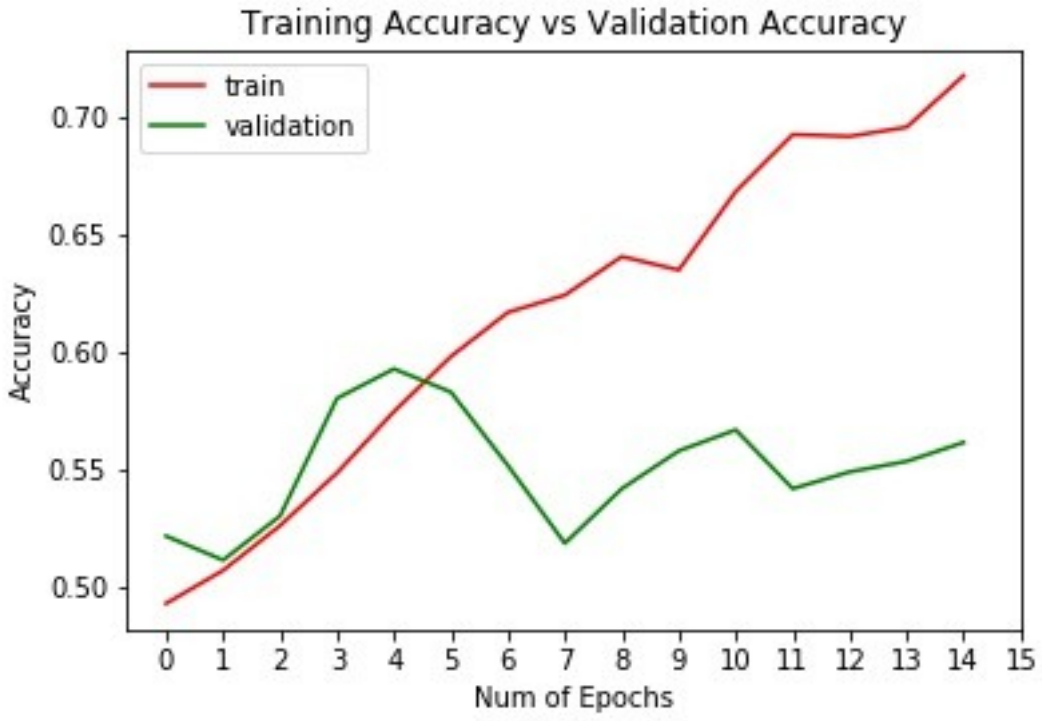


Figura 43: Entrenamiento con imágenes preprocesadas con ConvNet.

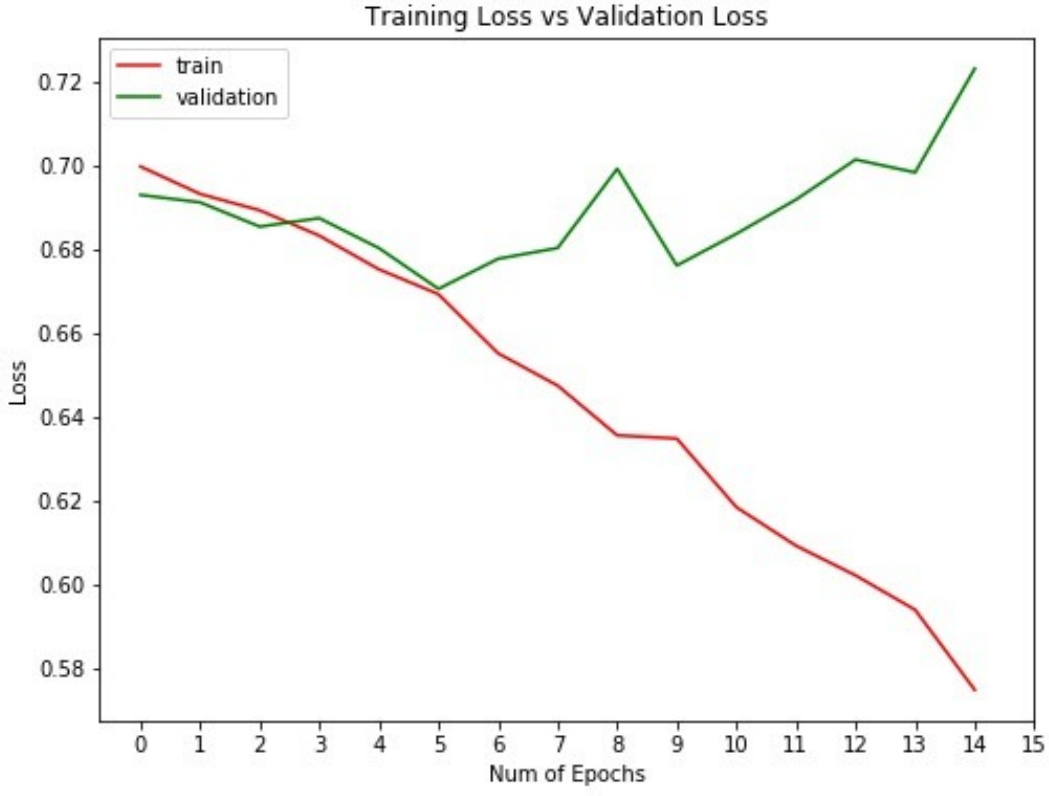


Figura 44: Entrenamiento con imágenes preprocesadas con ConvNet (pérdidas).

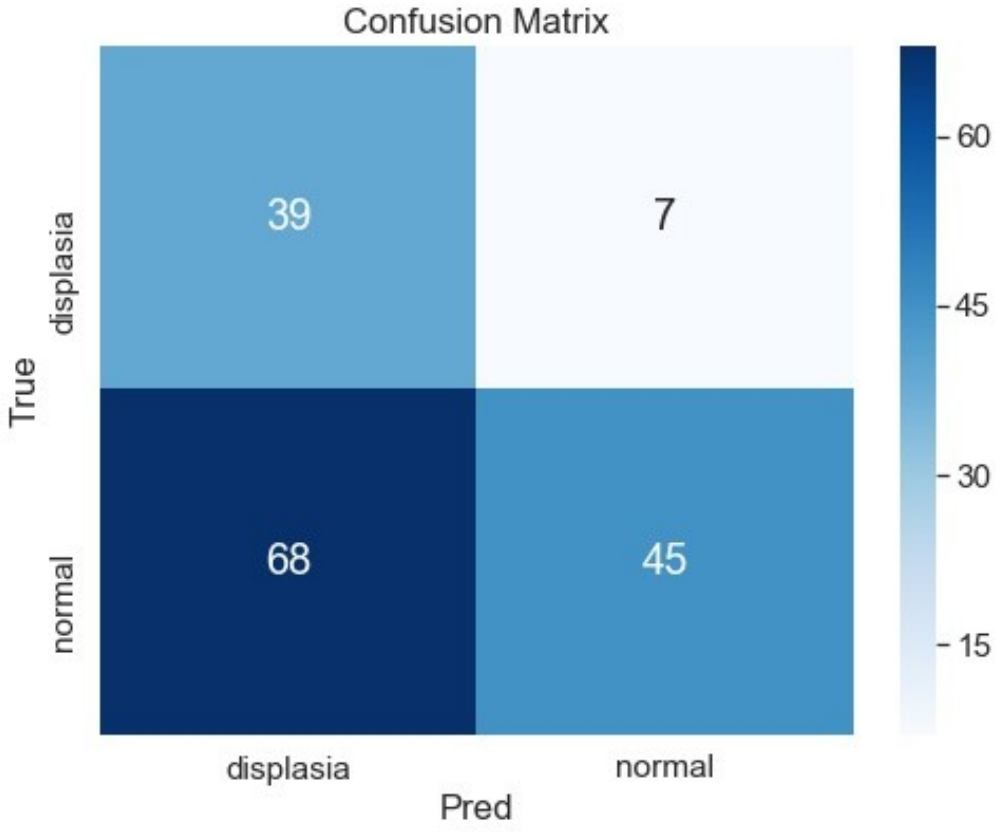


Figura 45: Matriz de confusión con la ConvNet y las imágenes preprocesadas.

Puede verse que la relación en imágenes que eran originalmente normales y son evaluadas como displásicas por la red es mayor que la de imágenes normales que la red clasifica como normales. Esto es un resultado cuestionable, pero debe decirse que los falsos positivos no son el peor de los resultados porque debe tenerse en cuenta que una aplicación con tendencia a los falsos positivos es revisada por un ojo experto para evitar tratar a personas sanas.

El resultado que sí es interesante es la baja tendencia a falsos negativos en las imágenes displásicas. En el conjunto de test, la red clasifica la mayoría de imágenes displásicas como tales, en un ratio de 39 a 7. De este resultado puede sacarse la buena lectura de que la red pasa por alto pocos casos en los que la displasia está presente.

Estos resultados dan idea de una cosa: se tienen pocas imágenes para entrenar redes que pudieran dar resultados aplicables en el entorno clínico en el estado actual. Sin embargo, son prometedores precisamente por la poca cantidad de muestras con las que se ha entrenado, consiguiendo a pesar de ello tendencias positivas y la experiencia suficiente para comenzar a generar un sistema de mayor eficacia.

6.3.3 Resultados con conjunto de imágenes de dermatología y la red convolucional sencilla

Con la idea de comprobar el funcionamiento del modelo se realizan pruebas de clasificación sobre un conjunto de datos de dermatología. En concreto son imágenes del mencionado challenge de ISIC. Con el mismo modelo se realizan entrenamientos para determinar el funcionamiento intentando clasificar una situación que es considerada como de mayor sencillez que las anoscopias. Se toman dos clases: imágenes que contienen nevus e imágenes que contienen carcinoma basocelular. Se toman imágenes en cantidades similares a las existentes de anoscopias y se

realizan entrenamientos con los siguientes resultados:

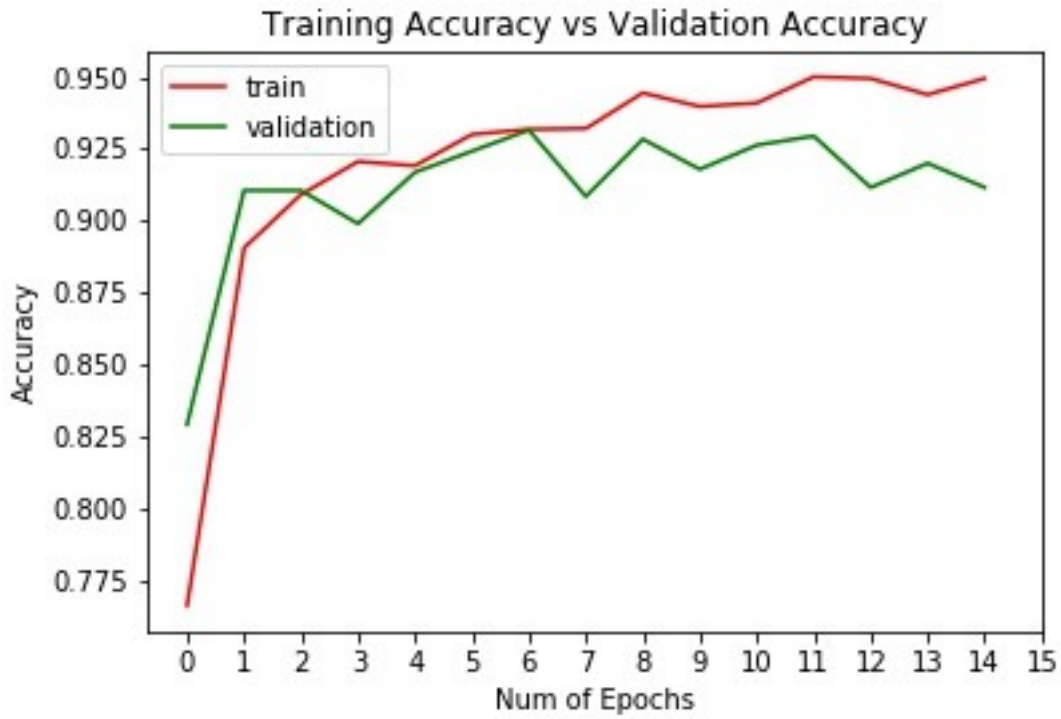


Figura 46: Precisión de entrenamiento en rojo y en verde de validación para la ConvNet con imágenes dermatológicas

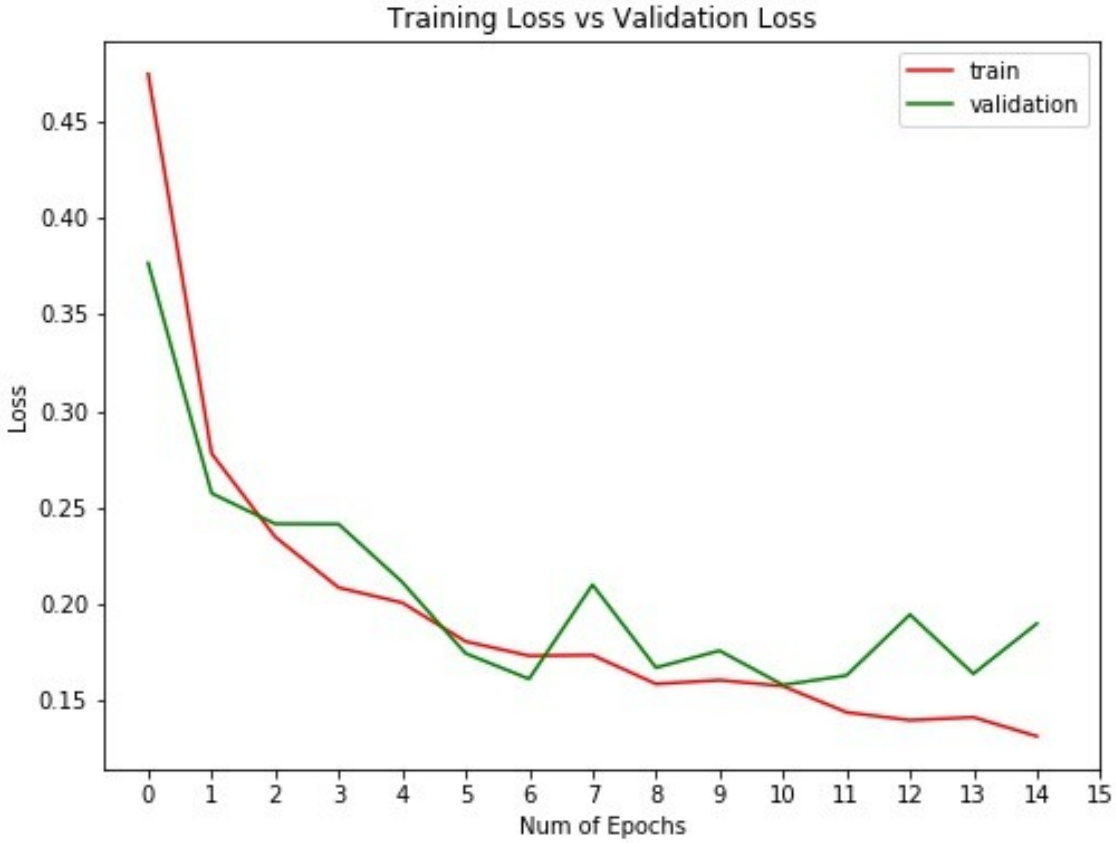


Figura 47: Evolución de las pérdidas de entrenamiento en rojo y en verde de validación para la ConvNet con imágenes dermatológicas

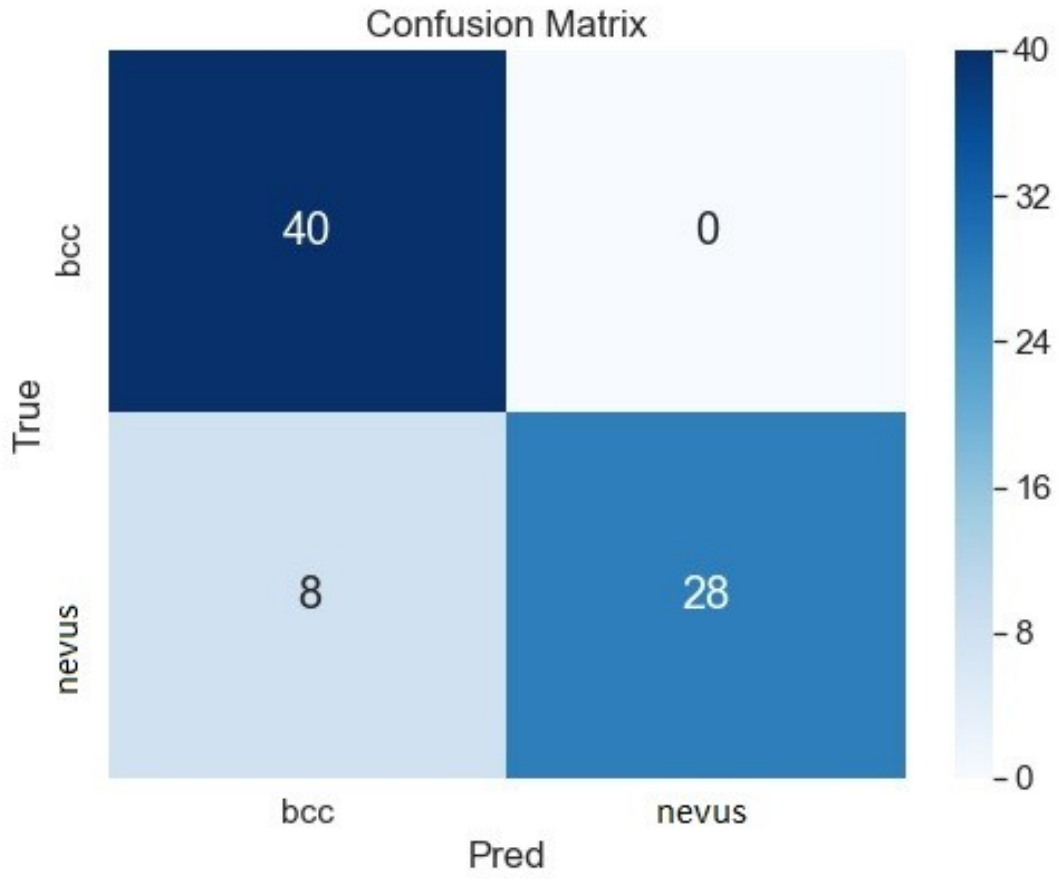


Figura 48: Matriz de confusión resultado de la clasificación nevus vs bcc

Como puede verse, el comportamiento de la red es el normal en un problema de clasificación. Con un conjunto de imágenes más fáciles de clasificar, la red tiene éxito, de manera que el modelo es capaz de resolver un problema basado en diferencias en imágenes.

Si el modelo es capaz de resolver un problema similar al de las anoscopias, de nuevo quizá aumentando la cantidad de imágenes sea posible encaminar esos resultados a los que presenta el problema de nevus contra carcinoma basocelular.

Existe la dificultad añadida de que se limita la toma de recortes de aquellas imágenes de anoscopia de las que se tiene resultados de biopsias. Cuando un profesional toma una biopsia lo hace únicamente si cree necesario, debido a una difícil decisión durante el diagnóstico acerca de la existencia de lesión precancerígena. Esto quiere decir que las biopsias de resultado normal han sido aquellas que mayor dificultad han presentado para el médico que la toma, aunque finalmente no hubiera lesión. Pero a la hora de clasificar en imágenes completas se necesitarán muchas más de tipo normal y muy evidentes para los profesionales para clasificarlas frente a las que resulten dudosas y las lesiones cancerosas.

7 CONCLUSIONES Y MEJORAS

En este último capitulo quiero dejar patente la idea de que este trabajo se ha realizado en el marco de un proyecto mayor, el denotado proyecto POMPIS. Éste, plantea algunos retos que en este trabajo no se han solventado o analizado en profundidad, de los cuales se intenta dejar algunas pinceladas al menos de aquello que podría resultar provechoso intentar.

Además y como es normal, se darán conclusiones y mejoras relativas al trabajo al que se ha dedicado mayor tiempo y esfuerzo.

7.1. Ortomacrofotografías: comentarios

La idea de la ortomacrofotografía es una parte del proyecto POMPIS que plantea la idea de reconstruir la línea escamo-columnar y generar una imagen que le de continuidad a ésta, de forma que pueda observarse la totalidad de la zona de interés a la vez. Esto no resulta posible en la práctica clínica habitual porque es una zona elástica y deformable.

La tarea de reconstrucción de la línea escamo-columnar podría ser más sencilla seleccionando fotogramas de una secuencia de vídeo, porque quizá garantizase una mayor cantidad de información acerca de la posición de la línea. Además, ya que la técnica que el profesional lleva a cabo le obliga a buscar la línea, habrá fotogramas suficientes si la anoscopia resulta exitosa. La parte que lo dificulta consiste en que, para introducir el anoscopio, se necesita cubrir la parte frontal del mismo con el mandil, de lo contrario se puede dañar la zona. Esto podría solventarse quizá si a cada fotograma que se de por bueno se le obliga a contener un círculo que coincida con el de la punta del anoscopio, de forma que mientras el profesional opera el aparato no lo podremos ver en las imágenes y éstas queden filtradas, inclusive, si la cámara se desenfoca, podría esto también filtrar esas imágenes borrosas al quedar el círculo del aparato peor definido. Una vez se analizase la secuencia, se podría intentar hacer el registro entre fotogramas consecutivos, en los que la diferencia será menor, facilitando la búsqueda de características locales comunes en las imágenes y la tarea de superposición de zonas, así como el suavizado en las partes de solapamiento entre las imágenes. Dificultades en las que se cae: discriminar zonas que hemos dado por buenas previamente si se recoge de nuevo esa zona con una perspectiva diferente, esto podría intentarse solucionar al ver cuál de las posibles superposiciones mejora una función de área por ejemplo, pensando en que si aumenta el área dentro de la línea escamo-columnar, será porque ésta se aleja más del centro de la región de interés, quedando así más definida y reconocible.

Se prueban diversas técnicas que buscan puntos afines en dos fotografías de forma que se intenta realizar la transformación de las imágenes para que se solapen. Esto resulta bastante complicado porque normalmente los médicos no toman muchas imágenes suficientemente parecidas a poca distancia unas de otras, y así, las transformaciones resultan muy complicadas, el registro de imagen médica es un tema de gran complejidad. Para tratar de comenzar a resolver esto, se comenta con los profesionales la posibilidad de realizar algunas anoscopias siguiendo unas directrices para la captura de fotografías de forma que se intente garantizar un mínimo de continuidad en la información disponible o la alternativa del uso de vídeos.

7.2. Automatización de la toma de recortes

Con los avances propuestos se está en la dirección correcta para llegar a automatizar la toma de muestras de las imágenes. En el momento en el que se localice el borde del anoscopio de cualquier imagen con suficiente acierto, si los profesionales realizan las descripciones clínicas siguiendo algún esquema, prácticamente se podría

comenzar la extracción automática con apenas algo más de trabajo.

En secciones previas se ha descrito una tabla de diagnóstico propuesta por los profesionales, que con su uso sería posible realizar en los textos digitalizados descritos por los médicos una búsqueda de palabras clave que describan la posición de la toma de biopsia. Junto con esto, se puede buscar el resultado de la biopsia en la fecha indicada y para el paciente en cuestión. Finalmente, localizado el borde del anoscopio, se podría: recortar el octante entero en el que la lesión está o realizar un recorte cuadrado (o con otra forma) en la posición comentada, intentando que abarque al menos el octante mencionado.

Esto que se describe es parte de lo que está previsto que sea desarrollado a modo de continuación de este trabajo. Como se ha mencionado, se va a continuar en la línea de aumentar y mejorar la base de datos disponibles para mejorar los resultados de las redes y conseguir de manera efectiva un algoritmo eficaz para los médicos.

7.3. Comentarios a la automatización de la clasificación de las imágenes

Durante el proceso de toma de fotografías, los médicos han seguido un *modus operandi*: al acudir el paciente a la clínica, se captura su identificador NHC y NUHSA y se realiza la toma de todas aquellas imágenes que sean relevantes, además de vídeos. Después se edita a mano el nombre de cada archivo, con la estructura explicada en el capítulo 5: identificador de paciente, seguido de la fecha de la intervención y luego viene una clasificación del médico, después hay una letra para diferenciar las imágenes de la misma intervención.

Se desea poder ofrecer a los médicos una solución que automatice esta labor. Por ejemplo: seleccionando todas las imágenes que pertenecen a una intervención para poder indicar el identificador y la fecha y además hacerlo con un programa con interfaz gráfica.

Esta pequeña aplicación les ahorrará a los profesionales tiempo y además evita casos en los que tras realizar una tarea repetitiva como esta se comentan fallos a la hora de trascribir ciertos identificadores o fechas. Otra ventaja es que se homogenizan los nombres de archivo, consiguiendo que la tarea posterior de extracción de características sea mucho mas sencilla.

7.4. Preprocesamiento

El caso de el ajuste de brillos es susceptible de una mejora clara. Podría ser interesante desarrollar un algoritmo que siga con la idea del propuesto pero con las siguientes mejoras:

- El umbralizado podría hacerse basado en el histograma, intentando ajustarse a la zona de valores altos tratando de optimizar la división de tonos entre brillos quemados y el resto de tonos naturales de los tejidos.

- Los tamaños de máscara de las operaciones morfológicas pueden ser optimizados realizando una detección de objetos de la imagen con los puntos brillantes detectados, para que fuera adaptándose a cada imagen de forma distinta. Podría hacerse un cálculo del área del objeto de mayor tamaño de entre las zonas de brillos para tener precisión y evitar que aquellos que son demasiado grandes se escapen del preprocesado.

- El proceso de rellenado de brillos es susceptible de mejora, ya que se realiza una interpolación que a veces resulta muy brusca a la vista y poco natural. Aunque se gane en brillos retirados, las aberraciones que acaban produciéndose deben intentar suavizarse y camuflarse mejor.

- El uso de mapas auto-organizados como parte del procesado del algoritmo resulta prometedor. No hay que insistir demasiado en que realizan un trabajo muy eficaz, los mapas de Kohonen son capaces de generar clústeres basados en colores y texturas que con el ajuste necesario, pueden llegar a ayudar enormemente a la red a la detección de patrones relevantes. Como desventaja del uso de los mapas, hay

que comentar que su trabajo resulta lento en un ordenador no especializado y procesar imágenes con resolución alta también resulta pesado computacionalmente.

El preprocesamiento es un punto clave que requiere de mucho mas esfuerzo de diseño puesto en él. En el momento en el que la red comienza a dar una dirección de mejora, debe dedicarse un gran esfuerzo en mejorar esas imágenes de forma que la precisión de la red crezca. Junto con el procesamiento, cuanto mayor sea la cantidad posible de muestras que se usen para entrenar, mayor se prevé que será la precisión, porque es así como la red trabaja con suficientes muestras como para generalizar la solución al problema.

En cuanto a la detección del borde se propone un sistema basado en los contornos activos que gestione el balance entre fuerzas internas y externas diferenciando de forma óptima con una curva las regiones más borrosas y las mejor definidas. Al encontrar esa curva, se busca que se obtenga en el borde del anoscopio, pudiendo así segmentar de manera eficaz el interior del exterior. Esto es útil para almacenar en cada ocasión la cantidad precisa de información. Además se consigue que todas las imágenes tengan el mismo aspecto.

La eliminación de brillos es bastante provechosa. Aunque la aplicación propuesta en la sección 5.6.1 resulta demasiado brusca y se nota bastante su procesado, los médicos no rechazan estos resultados. Es cierto que se crean ciertas aberraciones cromáticas y morfológicas porque el rellenado se basa en los valores de los bordes de los brillos, con lo cual si el brillo es un punto muy grande, se tiene mucha área rellenada únicamente con la referencia del borde, de forma que en superficies grandes se tiene una mancha de un color casi uniforme en el lugar del brillo. Resulta aceptable para los médicos porque el color del parche está directamente influenciado por el tono de las estructuras adyacentes.

El caso de los mapas auto-organizados da buenos resultados aunque a costa de un alto coste computacional. El uso de un ordenador mejor podría solventar esta desventaja pero no siendo así ha de denotarse. Obviando esto, el resultado del mapa auto-organizado es bastante bueno y está avalado por los trabajos relacionados que se han estudiado, como el caso de Simões *et al.,*2014.

Referido estrictamente a la librería utilizada: MiniSom debe decirse que resulta muy cómoda y fácil de utilizar. Sin embargo su intención minimalista puede conducir a que una adaptación al problema de dicha librería de mejor resultado, por ejemplo: rectificando los bloques saturados de zonas con brillos y asumiendo continuidad con los valores vecinos.

7.5. Comentarios al uso de distintos entornos de trabajo

El uso de diferentes entornos es enormemente provechoso. Es la forma de conseguir afianzar los métodos necesarios para realizar entrenamientos de redes neuronales. El trabajo en entornos variados provoca ajustar la realización de pruebas en favor de las capacidades disponibles en cada caso. Es necesario también afinar los parámetros de entrenamiento para salvar las limitaciones de los distintos sistemas.

Durante los entrenamientos en el ordenador portátil, resulta complejo para el equipo entrenar con imágenes de tamaños superiores a 150x150 – 300x300, dependiendo del modelo, su arquitectura y la cantidad de parámetros que la red necesita utilizar. Según la capacidad de la red se pueden emplear datos de mayor o menor tamaño. Los tiempos de ejecución resultan largos y esto provoca que se inutilice un sistema del que existe dependencia para el desarrollo de otras tareas porque los entrenamientos consumen gran parte de la capacidad computacional del equipo y éste debe permanecer activo todo el tiempo. Además, la necesidad de reducir el tamaño de los datos hace que en el caso de las imágenes, la búsqueda de detalles pequeños se torna mas complicada.

7.6. ¿La red puede mejorar?

Sí, la red aún tiene mucho margen de mejora. Los algoritmos que se utilizan como solución comercial están situados en niveles de precisión de entorno al 90% (en términos generales, hay tareas en las que resulta aceptable

un índice de precisión menor). Esta cifra aún puede resultar baja en el caso de la medicina, donde casi no se cede la toma de decisiones aún a máquinas por precisas que sean.

En ese sentido, el estado actual del algoritmo no debe quizá entenderse como malo. El hecho de haber comenzado a dar una tendencia positiva es seña de que con muchos más recortes y un procesamiento severo se puede llegar mejoras en la precisión. Los resultados expuestos dejan ver que es necesaria una mejora en la toma de datos, para que el trabajo de clasificación goce de mínimas garantías y homogeneidades.

La red puede dar buenos resultados también con el uso de modelos mejores. Los modelos que se han utilizado están basados fuertemente en arquitecturas de uso general, que tienen un éxito contrastado. Pero es posible que el diseño de un modelo más adaptado al problema que aquí se trata no resulte excesivamente costoso y sí sea notable una mejora.

En los modelos usados de gran capacidad, se ha obtenido un sobreajuste notable, haciendo patente la necesidad de utilizar más datos, así que el uso de modelos de menor capacidad es una solución para los conjuntos de muestras disponibles.

7.7. ¿Cuál es el alcance del proyecto en un futuro?

En un estado posterior al que se tiene al acabar este trabajo, sería interesante conseguir combinar el algoritmo con capacidad de predecir satisfactoriamente las lesiones con la aplicación ImageCropper. Así, un médico que se encuentre entrenando podría ir viendo la evolución de los pacientes y comprobar si tiene dudas en algún caso qué predicción lanza la red al hacer click sobre una zona sospechosa, de forma similar a como se extraen los recortes.

Es ambicioso, pero la implementación de la red neuronal en un sistema de extracción de recortes de imágenes suficientemente rápido podría utilizarse en tiempo real mientras se hace un análisis clínico. Esto convertiría al proyecto en un sistema real de ayuda al diagnóstico. Podría servir de apoyo también en el aprendizaje de médicos inexpertos que se decidan a realizar HRA.

Idealmente se abogaría por conseguir que el algoritmo fuera capaz de clasificar una imagen completa comprobando recortes de la misma y que luego reconstruyera para toda la imagen una máscara de las zonas examinadas diferenciadas según la clase asignada durante la predicción. Así se realizaría una especie de clasificación/segmentación semántica.

Debido a que es un área de la medicina que no es objeto de demasiados estudios, como debe haberse notado en el estado del arte, no hay casos de referencias a estudios de las lesiones previas al cáncer anal con técnicas de *deep learning*, así que el presente trabajo puede servir como un buen punto de partida para comenzar a trabajar en una herramienta que, de llegar a ser eficaz podría llegar a ofrecerse a toda la comunidad. Con un poco de esa ambición, y a la vista de que el proyecto avanza, se ofrece desde FISEVI a quien escribe estas líneas la posibilidad de intentarlo, para continuar trabajando en el entorno del HUVR. Esto garantiza la continuidad del proyecto y que siga creciendo, y ojalá que lo haga siempre en la dirección en la que el gradiente señala el mínimo de la función de pérdidas.

REFERENCIAS

[1] P. Viciana *et al.*, "Proyecto de Ortofotografía Macroscópica Para Imágenes Sistemáticas de displasia canal anal (POMPIS _ analcanal)," pp. 1–24, 2019.

[2] P. W. Simões *et al.*, "Classification of images acquired with colposcopy using artificial neural networks," *Cancer Inform.*, vol. 13, pp. 119–124, 2014.

[3] S.V. Graham, "The human papillomavirus replication cycle, and its links to cancer progression: a comprehensive review," *Clinical Science,* vol. 131, pp 2201-2221, 2017

[4] International Agency for Research on Cancer Working Group, "Biological Agents: A review of human carcinogens. Human Papillomaviruses," *IARC Monographs on the evaluation of carcinogenic risks to humans*, vol 100 B,2012

[5] S.E. Goldstone, "Diagnosis and Treatment of HPV-Related Squamous Intraepithelial Neoplasia in Men Who Have Sex with Men," *Physicians' Research Network*, vol 10, número 4,2005

[6] D. De la Fuente-Villarreal *et al.*, "Biología del Virus del Papiloma Humano y técnicas de diagnóstico," *Medicina Universitaria,* vol. 12(49), pp. 231-238,2010

[7] R.J. Hillman *et al.,*"2016 IANS International Guidelines for Practice Standards in the Detection of Anal Cancer Precursors," *Journal of Lower Genital Tract Disease*, vol 20 (4) pp-283-291,2016

[8] M. Camus *et al.,*"Which Lesions Should Be Biopsied During High-Resolution Anoscopy? Prospective Descriptive Study of Simple Morphological Criteria," *Journal of Lower Genital Tract Disease, vol 19(2), 2015*

[9] D. Machalek *et al.*,"Anal human papillomavirus infection and associated neoplastic lesions in men who have sex with men: a systematic review and meta-analysis," *Lancet Oncol,* vol 13, pp. 487-500, 2012

[10] J. Palefsky, "Human papillomavirus infection and its role in the pathogenesis of anal cancer", *Seminars in Colon and Rectal Surgery,* vol 28, pp. 57-62

[11] E.J. Topol, "High-performance medicine: the convergence of human and artificial intelligence," *Nature Medicine,* vol 25, pp. 44-56 , 2019

[12] Y. Jussman *et al.,*"Intelligent Screening Systems for Cervical Cancer"

[13] Y. LeCun *et al.,* "Deep learning" *Nature,* vol 521, pp436-444 , 2015

[14] I. Goodfellow *et al.,* "Deep Learning",*MIT Press*, 2016

[15] G. Litjiens *et al.,* "A survey on deep learning in medical image analysis"

[16] T. Kohonen , "The Self-Organizing Map" *Proceedings of the IEEE*, vol 78(9), 1990

[17] I. Claude *et al.,*"Contour features for colposcopic image classification by artificial neural networks", *Object recognition supported by user interaction for service robots*, vol 1, pp. 771-774, 2002

[18] L.G. Koss *et al,* "Significant Reduction in the Rate of False-Negative Cervical Smears With Neural Network-Based Technology (PAPNET Testing System)", *Human Pathology*, vol. 28 (10) pp. 1196-1203

[19] A. Shrestha y A. Mahmood, "Review of Deep Learning Algorithms and Architectures", *IEE Access*, vol 7, pp. 53040-53065, 2019

[20] J. Schmidhuber, "Deep learning in neural networks: An overview" *Neural Networks*, vol 61, pp. 85-117, 2014

[21] J. Patterson y A. Gibson, "Deep Learning A Practitioner's Approach", *O'Reilly*, 2017

Recursos web

Web del challenge científico propuesto por ISIC:

https://challenge2019.isic-archive.com/

TensorFlow Paper de 2015:

M. Abadi *et al.,*"TensorFlow:Large-Scale Machine Learning on Heterogeneous Distributed Systems ", https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf

Resultados ImageNet 2014:

http://www.image-net.org/challenges/LSVRC/2014/results

Desarrollo del uso de Inception v3:

https://cloud.google.com/tpu/docs/inception-v3-advanced

Recursos y tutoriales de deeplearning.net:

https://deeplearning.net/

Web de Keras, con documentación:

https://keras.io/

Documentación de Python:

https://www.python.org/doc/

Web con similitudes entre el uso de la librería numpy de Python y MatLab:

https://docs.scipy.org/doc/numpy/user/numpy-for-matlab-users.html

Documentación de numpy:

https://numpy.org/doc/1.17/reference/index.html

Documentación de matplotlib:

https://matplotlib.org/3.1.1/contents.html

Documentación de tkinter:

https://docs.python.org/2/library/tkinter.html#tkinter-modules

Documentación de tkinter (otra versión):

https://docs.python.org/3/library/tk.html

Guía tkinter en español:

https://guia-tkinter.readthedocs.io/es/develop/chapters/1-intro/1.0-Intro.html

Tutorial/documentación de OpenCV:

https://docs.opencv.org/4.1.2/d6/d00/tutorial_py_root.html

https://docs.opencv.org/ (todas las versiones)

Web con guías y lugar para compartir códigos con la comunidad:

https://stackoverflow.com/

Anexo

A continuación se incluye separado en bloques lo mas funcionales que ha sido posible el código completo y las funciones necesarias para el funcionamiento de la aplicación ImageCropper. La definición __init__ es la que realiza la inicialización de las variables, la generación de las rejillas para disponer los elementos y pone a funcionar algunas funciones como el visor de imágenes.

```
1.  import tkinter as tk
2.  from tkinter import filedialog,ttk
3.  from PIL import Image, ImageTk, ImageGrab
4.  import os
5.  import cv2
6.  import numpy as np
7.
8.
9.  class imageCropper(tk.Frame):
10.
11.     patrones =
['Marmolado','Valecula','Bordes','PVascularArrosariado','PVascularPunteado','PVascularLong','PVascul
arMosaicismo','MicroPapilas','Elevado','Verrugas','Condiloma','AIN1','AIN2','AIN3','Plano']
12.     contrast = ['Acetico','Aceticosucio','LugolParcial','LugolPositivo','LugolNegativo']
13.     octantes = ['','Oct1','Oct2','Oct3','Oct4','Oct5','Oct6','Oct7','Oct8']
14.     patrones2=
['','ABPalido','ABGris','MargenPreciso','MargenImpreciso','ContornoPlano','ContornoElevado','Contorn
oMicroPapilas','VasosLong','VasosPunteado','VasosEstriados','VasosMosaico','LugolNegativo','LugolPar
cial','LugolPositivo','NBiopsia']
15.     resBiopsias=['cond','norm','AIN1','AIN2','AIN3']
16.
17.     out_paths  = ['Biopsias con Resultado Condiloma','Biopsias Con Resultado Normal','Biopsias
con Resultado AIN1', 'Biopsias con resultado AIN2', 'Biopsias con resultado AIN3']
18.
19.
20.
21.     def selector(self):
22.         self.parametro_patron=self.var_boton.get()
23.
24.     def selector2(self):
25.         self.cadenaNombrePatrones = str('')
26.         for j in range(len(self.lista_selectores)):
27.             print('ese no era 0',self.lista_selectores[j].get())
28.             self.cadenaNombrePatrones += self.patrones2[self.lista_selectores[j].get()]
29.             print(self.cadenaNombrePatrones)
30.             if(self.lista_selectores[j].get() is not 0 and
self.patrones2[self.lista_selectores[j].get()] in self.cadenaNombrePatrones):
31.                 self.cadenaNombrePatrones += str('_')
32.
33.     def selector3(self):
```

```
        self.cadenaNombreOctantes = str('')
        for j in range(len(self.lista_octantes)):
            print('ese no era 0',self.lista_octantes[j].get())
            self.cadenaNombreOctantes += self.octantes[self.lista_octantes[j].get()]
            print(self.cadenaNombreOctantes)
            if(self.lista_octantes[j].get() is not 0 and
self.octantes[self.lista_octantes[j].get()] in self.cadenaNombreOctantes):
                self.cadenaNombreOctantes += str('_')

    def selector4(self):
        self.cadenaNombreBiopsias = str('')
        for j in range(len(self.resBiopsias)):
            if(self.varb0.get() != 0):
                self.cadenaNombreBiopsias = self.resBiopsias[self.varb0.get()] + str('_')
                print(self.cadenaNombreBiopsias)
            self.out_path2 =
os.path.join('HPV_imagen','recortes',self.out_paths[self.varb0.get()])
            self.recortesTotal = os.listdir(self.out_path2)
            self.out_path  = self.out_path2
            print(self.out_path2)

    def buscaID(self,images,total):
        id_paciente_len= [0 for b in range(total)];
        id_num         = 0;
        for i in range(total):
            id_num=0
            #print(images[i])
            for j in range(1,len(images[i])):
                #print('valor i:',int(i),'  valor j:',j)
                if(images[i][j] >='0'and images[i][j] <= '9'):
                    id_num = id_num + 1;
                    #print(images[i][j+1] > '9','')
                if(images[i][j+1] < '0' or images[i][j+1] > '9' ):
                    #print('Ultimo caracter: ',images[i][id_num-1] )
                    id_paciente_len[i]=id_num+2  #la primera vez que dejemos de ver numeros,
termina el id de paciente
                    #print(images[i][0:id_num+2])
                    break
        self.long_ids = id_paciente_len

    def coincide(self,total):
        nombres = np.array(self.images)
        coincide= np.zeros((total,1))
        id_paciente_len=self.long_ids
        print(nombres.shape)
        for i in range(total-1):
            for j in range(total-1):
                if(nombres[i][0:id_paciente_len[i]]==nombres[j+1][0:id_paciente_len[j+1]]):
#coincidencia del codigo de paciente
                    coincide[j+1]=i;
        #print(nombres[4][0:id_num])
        coincide[0]=coincide[1]  #voy a suponer aqui que la primera imagen no es la unica que
tenemos de ese paciente
        self.coincidencias=coincide


    def cropCrea(self,posx,posy,tam,img):    #acepta 3 ints de posicion del recorte y tamaño y
una imagen leida por opencv
        hei,wid,chan=img.shape
```

```
87.         if(posx-tam/2 <= 0):
88.             posx = tam/2
89.             print('izq')
90.         if(posx+tam/2 >=wid):
91.             posx=wid-tam/2-1
92.             print('der')
93.         if(posy-tam/2 <=0):
94.             posy=tam/2
95.             print('arr')
96.         if(posy+tam/2 >=hei):
97.             posy=hei-tam/2-1
98.             print('aba')
99.         print("Coordenadas:posx,posy",posx,posy)
100.            self.posx = posx
101.            self.posy = posy
102.            self.tam  = tam
103.            self.hei  = hei
104.            self.wid  = wid
105.            self.chan = chan
106.            print("Tam imagen :",str([posy-tam/2,posy+tam/2,posx-tam/2,posx+tam/2]))
107.            return img[round(posy-tam/2):round(posy+tam/2),round(posx-tam/2):round(posx+tam/2)]
108.
109.
110.        def damePath(self):
111.            self.out_path = filedialog.askdirectory(title='Please select a directory')
    #directorio para imagenes usadas en recortes
112.
113.        def damePath2(self):
114.            self.out_path2 = filedialog.askdirectory(title='Please select a directory')
#directorio para recortes
115.            self.recortesTotal = os.listdir(self.out_path2)
116.
117.        def damePath3(self):
118.            self.data_path = filedialog.askdirectory(title='Por favor, seleccionar el directorio
de entrada de imagenes')
119.            self.images    = os.listdir(self.data_path)
120.            self.buscaID(self.images,len(self.images))
121.            self.cambia_path=True
122.            self.siguienteImagen()
123.
124.        def callback(self,event):
125.            tam=128
126.            posx=event.x
127.            posy=event.y
128.            #cropped = Image.fromarray(cv_img).crop((250, 150, 40, 40))
129.

cv_img=cv2.cvtColor(cv2.imread(os.path.join(self.data_path,self.images[self.cuenta]),1),cv2.COLOR_BG
R2RGB)
130.            cv_img = cv2.resize(cv_img,(self.ancho2,self.alto2))
131.            cropped = self.cropCrea(posx,posy,tam,cv_img)
132.            self.cropeada= cropped
133.            #cropped = self.imcrop(cv_img,(posy-tam),(posy+tam),(posx-tam))
134.            print(cropped.shape)
135.            print(posx-tam/2,posy-tam/2)
136.            print(posx-tam, posx+tam,posy+tam,posy-tam)
137.            self.root.photo2 = ImageTk.PhotoImage(Image.fromarray(cropped))
138.            height,width,canales=cropped.shape
139.            print(height,width)
```

```
140.            imagcrop=self.cvas2.create_image(tam/2,tam/2,image=self.root.photo2)
141.            #print(self.cropeada)
142.            self.update_idletasks()
143.
144.        def actualizaCuenta(self,event):
145.            self.cuenta=int(self.dato_entrada.get()-1)
146.            self.siguienteImagen()
147.            print("Entrada.get :"+str(self.cuenta))
148.
149.        def __init__(self,parent,*args,**kwargs):
150.            var = tk.IntVar()
151.            self.var0 = tk.IntVar()
152.            self.var1 = tk.IntVar()
153.            self.var2 = tk.IntVar()
154.            self.var3 = tk.IntVar()
155.            self.var4 = tk.IntVar()
156.            self.var5 = tk.IntVar()
157.            self.var6 = tk.IntVar()
158.            self.var7 = tk.IntVar()
159.            self.var8 = tk.IntVar()
160.            self.var9 = tk.IntVar()
161.            self.var10= tk.IntVar()
162.            self.var11= tk.IntVar()
163.            self.var12= tk.IntVar()
164.            self.var13= tk.IntVar()
165.            self.var14= tk.IntVar()
166.            self.lista_selectores = [self.var0 , self.var1, self.var2, self.var3,
167.                                     self.var4, self.var5, self.var6, self.var7,
168.                                     self.var8, self.var9, self.var10, self.var11,
169.                                     self.var12, self.var13, self.var14]
170.
171.            self.var00 = tk.IntVar()
172.            self.var01 = tk.IntVar()
173.            self.var02 = tk.IntVar()
174.            self.var03 = tk.IntVar()
175.            self.var04 = tk.IntVar()
176.            self.var05 = tk.IntVar()
177.            self.var06 = tk.IntVar()
178.            self.var07 = tk.IntVar()
179.            self.lista_octantes = [self.var00 , self.var01, self.var02, self.var03 ,
180.                                     self.var04 , self.var05 , self.var06 , self.var07]
181.
182.            self.varb0 = tk.IntVar()
183.
184.            self.cadenaNombrePatrones = ''
185.            tk.Frame.__init__(self, parent, *args, **kwargs)
186.            self.root = parent
187.            self.parent=parent
188.            self.root.wm_title("ImageCropper")
189.            #self.root.geometry('1500x1000')
190.
191.            self.cuenta     = 0
192.
193.
194.            #hace falta ahora p.e. elegir un directorio
195.            self.folder_path='100m1_nuhsa'
196.            self.data_path='HPV_imagen'
```

```
197.            self.data_path = os.path.join(self.data_path,self.folder_path)  #set0: '', set1:
i100m1, set2: i200m1, set3: i300m1
198.            self.images    = os.listdir(self.data_path)
199.            #self.images    = np.loadtxt('imagenes_grado_3.txt',dtype='str')
200.            #self.images     =
np.genfromtxt('imagenes_grado_3_i200_nuhsa.txt',skip_header=0,dtype='str')
201.            total = len(self.images)
202.            self.buscaID(self.images,total)
203.            print('Long id foto 1: ',self.long_ids[1])
204.            imagenes=np.array(self.images)
205.            print('Total de imagenes de test:',total)
206.            print(self.images[0][0:])
207.            input_image =
cv2.cvtColor(cv2.imread(os.path.join(self.data_path,self.images[self.cuenta]),1),cv2.COLOR_BGR2RGB)
208.            self.alto = round(input_image.shape[0] / 4)
209.            self.ancho= round(input_image.shape[1] / 4)
210.
211.            #frames y lienzos (hay 2, un visualizador principal y el visor de recorte)
212.
213.            self.frame1 = tk.Frame(self.root, width=self.ancho, height=self.alto,bd=2)
214.            self.frame1.grid(row=0, column=0,rowspan=4,columnspan=4)
215.            self.cv1 = tk.Canvas(self.frame1, height=self.alto, width=self.ancho,
background="white", bd=2, relief=tk.RAISED)
216.            self.cv1.grid(row=0,column=0)
217.
218.            self.frame2 = tk.Frame(self.root, width=128, height = 128, bd=1)
219.            self.frame2.grid(row=2,column=4)
220.            self.cvas2 = tk.Canvas(self.frame2, height=128, width=128, bd=1,relief=tk.SUNKEN)
221.            self.cvas2.grid(row=2,column=4)
222.
223.            #botoneras y selectores
224.
225.            broButton = tk.Button(self.root, text='Siguiente', height=2, width=8, command =
self.siguienteImagen)
226.            broButton.grid(row=0, column=4, padx=2, pady=2)
227.
228.            broButton2 = tk.Button(self.root, text='Anterior', height=2, width=8, command =
self.previaImagen)
229.            broButton2.grid(row=1, column=4, padx=2, pady=2)
230.
231.            cropButton = tk.Button(self.root, text='Guardar Recorte', height=2, width=14, command
= self.guardaCrop) #hay que inicializar photo2?
232.            cropButton.grid(row=3, column=4, padx=2, pady=0)
233.
234.            pathButton = tk.Button(self.root, text='Elige directorio para imagenes ', height=2,
width=22, command = self.damePath)
235.            pathButton.grid(row=4, column=1, padx=2, pady=2,sticky='w')
236.
237.            pathButton1 = tk.Button(self.root, text='Elige directorio para recortes', height=2,
width=22, command = self.damePath2)
238.            pathButton1.grid(row=4, column=2, padx=2, pady=2,sticky='w',columnspan=1)
239.
240.            pathButton2 = tk.Button(self.root, text='Elige directorio de entrada', height=2,
width=22, command = self.damePath3)
241.            pathButton2.grid(row=4, column=0, padx=2, pady=2,sticky='w')
242.
243.            botontexto1 = tk.Button(self.root, text='+', height=2, width=2, command =
self.textoSiguiente)
244.            botontexto1.grid(row=5,column=4,padx=2,pady=2,sticky='sw')
245.
```

```
246.            botontexto2 = tk.Button(self.root, text='-', height=2, width=2, command =
self.textoAnterior)
247.            botontexto2.grid(row=6,column=4,padx=2,pady=2,columnspan=1,sticky='nw')
248.
249.            botontexto3 = tk.Button(self.root, text='+', height=2, width=2, command =
self.textoSiguiente2)
250.            botontexto3.grid(row=5,column=5,padx=2,pady=2,sticky='sw')
251.
252.            botontexto4 = tk.Button(self.root, text='-', height=2, width=2, command =
self.textoAnterior2)
253.            botontexto4.grid(row=6,column=5,padx=2,pady=2,columnspan=1,sticky='nw')
254.
255.
256.            #selectores de octantes
257.            r00 = ttk.Checkbutton(self.root,text=self.octantes[1],variable=self.var00,
command=self.selector3, onvalue=1)
258.            r01 = ttk.Checkbutton(self.root,text=self.octantes[2],variable=self.var01,
command=self.selector3, onvalue=2)
259.            r02 = ttk.Checkbutton(self.root,text=self.octantes[3],variable=self.var02,
command=self.selector3, onvalue=3)
260.            r03 = ttk.Checkbutton(self.root,text=self.octantes[4],variable=self.var03,
command=self.selector3, onvalue=4)
261.            r04 = ttk.Checkbutton(self.root,text=self.octantes[5],variable=self.var04,
command=self.selector3, onvalue=5)
262.            r05 = ttk.Checkbutton(self.root,text=self.octantes[6],variable=self.var05,
command=self.selector3, onvalue=6)
263.            r06 = ttk.Checkbutton(self.root,text=self.octantes[7],variable=self.var06,
command=self.selector3, onvalue=7)
264.            r07 = ttk.Checkbutton(self.root,text=self.octantes[8],variable=self.var07,
command=self.selector3, onvalue=8)
265.            r00.grid(row=2,column=5,padx=2,pady=2)
266.            r01.grid(row=2,column=6,padx=2,pady=2)
267.            r02.grid(row=2,column=7,padx=2,pady=2)
268.            r03.grid(row=2,column=8,padx=2,pady=2)
269.            r04.grid(row=2,column=9,padx=2,pady=2)
270.            r05.grid(row=2,column=10,padx=2,pady=2)
271.            r06.grid(row=2,column=11,padx=2,pady=2)
272.            r07.grid(row=2,column=12,padx=2,pady=2)
273.
274.            #selectores de patron de enfermedad (generar tras un evento?,para que el usuario
pueda añadir autónomamente)
275.            r1=ttk.Checkbutton(self.root,text=self.patrones2[1],variable=self.var0,
command=self.selector2, onvalue=1)
276.            r2=ttk.Checkbutton(self.root,text=self.patrones2[2],variable=self.var1,
command=self.selector2, onvalue=2)
277.            r3=ttk.Checkbutton(self.root,text=self.patrones2[3],variable=self.var2,
command=self.selector2, onvalue=3)
278.            r4=ttk.Checkbutton(self.root,text=self.patrones2[4],variable=self.var3,
command=self.selector2, onvalue=4)
279.            r5=ttk.Checkbutton(self.root,text=self.patrones2[5],variable=self.var4,
command=self.selector2, onvalue=5)
280.            r6=ttk.Checkbutton(self.root,text=self.patrones2[6],variable=self.var5,
command=self.selector2, onvalue=6)
281.            r7=ttk.Checkbutton(self.root,text=self.patrones2[7],
variable=self.var6,command=self.selector2, onvalue=7)
282.            r8=ttk.Checkbutton(self.root,text=self.patrones2[8],
variable=self.var7,command=self.selector2, onvalue=8)
283.            r9=ttk.Checkbutton(self.root,text=self.patrones2[9],
variable=self.var8,command=self.selector2, onvalue=9)
284.            r10=ttk.Checkbutton(self.root,text=self.patrones2[10],
variable=self.var9,command=self.selector2, onvalue=10)
285.            r11=ttk.Checkbutton(self.root,text=self.patrones2[11],
variable=self.var10,command=self.selector2, onvalue=11)
286.            r12=ttk.Checkbutton(self.root,text=self.patrones2[12],
```

```
variable=self.var11,command=self.selector2, onvalue=12)
            r13=ttk.Checkbutton(self.root,text=self.patrones2[13],
variable=self.var12,command=self.selector2, onvalue=13)
            r14=ttk.Checkbutton(self.root,text=self.patrones2[14],
variable=self.var13,command=self.selector2, onvalue=14)

            #posicionamiento de los selectores
            r1.grid(row=3,column=5,padx=2,pady=2)
            r2.grid(row=3,column=6,padx=2,pady=2)
            r3.grid(row=3,column=7,padx=2,pady=2)
            r4.grid(row=3,column=8,padx=2,pady=2)
            r5.grid(row=3,column=10,padx=2,pady=2)
            r6.grid(row=3,column=11,padx=2,pady=2)
            r7.grid(row=3,column=12,padx=2,pady=2)
            r8.grid(row=4,column=5,padx=2,pady=2)
            r9.grid(row=4,column=6,padx=2,pady=2)
            r10.grid(row=4,column=7,padx=2,pady=2)
            r11.grid(row=4,column=8,padx=2,pady=2)
            r12.grid(row=4,column=10,padx=2,pady=2)
            r13.grid(row=4,column=11,padx=2,pady=2)
            r14.grid(row=4,column=12,padx=2,pady=2)

            #selectores de resultado, llaman a una función y cambian out_path para que se guarde
la estructura de directorios
            #ademas agrega el resultado de la biopsia al nombre del archivos
            rb1 = ttk.Radiobutton(self.root,text='Condiloma',variable = self.varb0, command =
self.selector4, value=0)
            rb2 = ttk.Radiobutton(self.root,text='Normal',variable = self.varb0, command =
self.selector4, value=1)
            rb3 = ttk.Radiobutton(self.root,text='AIN1',variable = self.varb0, command =
self.selector4, value=2)
            rb4 = ttk.Radiobutton(self.root,text='AIN2',variable = self.varb0, command =
self.selector4, value=3)
            rb5 = ttk.Radiobutton(self.root,text='AIN3',variable = self.varb0, command =
self.selector4, value=4)

            rb1.grid(row=1, column = 5, padx=2,pady=2)
            rb2.grid(row=1, column = 6, padx=2,pady=2)
            rb3.grid(row=1, column = 7, padx=2,pady=2)
            rb4.grid(row=1, column = 8, padx=2,pady=2)
            rb5.grid(row=1, column = 9, padx=2,pady=2)


            # self.var2 = tk.IntVar()
            # la tabla comienza desde el punto del grid 3,5 (+1,+1 para agregar los nombres de
las celdas de la tabla) hasta el 18 13
            # for k in range(len(self.octantes)):
                # for j in range(len(self.patrones2)):
                    # ch=tk.Checkbutton(root, variable = self.var2   , command= self.selector)
                    # ch.grid(row=2+k,column=5+j)
            #se agrega un cuadro de texto para leer la evolucion de la intervencion de dicho
paciente y otro cuadro que destaque el nombre de la imagen que se ver
            self.T = tk.Text(root,height=1,width=30)
            self.T.grid(row=4,column=3,padx=2,pady=2)

            self.T2 = tk.Text(root,height=40,width=100)
            self.T2.grid(row=5,column=0,rowspan=4,columnspan=4,padx=2,pady=2,sticky='w')
            #otro cuadro de texto para mostrar la imagen actual y la total de la lista

            self.T3 = tk.Text(root,height=1,width=12)
```

```
336.            self.T3.grid(row=0,column=5)
337.
338.            #otro cuadro de texto para la informacion de las biopsias
339.            self.T4 = tk.Text(root,height=30,width = 80)
340.            self.T4.grid(row=6,column=5, rowspan=3, columnspan=10)
341.
342.            #Carga de archivos e inicializacion de vbles
343.
344.            self.evolucion= np.genfromtxt('hojas4.csv',skip_header=1,delimiter=';;',dtype='str')
345.            self.evolucion_paciente = ['Datos de evolucion de paciente']
346.            self.total2 = self.evolucion.shape
347.            self.bandera = 0
348.
349.            #Carga de datos de biopsias
350.            self.biopsias=
np.genfromtxt('hojabiopsiasNUHSA.txt',dtype='str',delimiter=';;',encoding="utf8",skip_header=1)
351.            self.biopsiaPaciente = []
352.            self.total3 = self.biopsias.shape[0]
353.            print(self.total3, self.biopsias[0])
354.
355.            #self.coordenadas = ['Coordenadas: posicion x,posicion y,tamanho,height
(imagen),width (imagen),chan (canales)']
356.            self.coordenadas = []
357.            self.cuentaevototal=0
358.            self.coincide(total)
359.            self.cambia_path = False
360.            #print(var)
361.            self.var_boton=var
362.            self.repite_paciente=0
363.            self.cuenta_anterior=0
364.            self.max_cuenta = total-1
365.
366.
367.
368.            self.dato_entrada  = tk.IntVar()
369.            etiqueta1     = tk.Label(root, textvariable = self.dato_entrada)
370.            etiqueta1.grid(row=0,column=6)
371.            entrada_texto = tk.Entry(root,textvariable=self.dato_entrada)
372.            entrada_texto.grid(row=0,column=6)
373.    #       dato_entrada.trace("w",self.actualizaCuenta(dato_entrada.get()))
374.            entrada_texto.bind("<Return>",self.actualizaCuenta)
375.            #boton5        = tk.Button(self.root, text='Ir a imagen', height=2, width=5, command
= self.actualizaCuenta())
376.            #boton5.grid(row=6,column=4,padx=2,pady=2,columnspan=1,sticky='nw')
377.
378.            self.siguienteImagen()     #se llama aqui para que se muestre la primera
379.
380.            #se crea una entrada de texto para cambiar el numero de imagen actual y no tener que
ir de 1 en 1
381.
382.
383.        def siguienteImagen(self):
384.            #print(self.data_path)
385.            #print(os.path.join(self.data_path,self.images[self.cuenta]))
386.            if(self.coincidencias[self.cuenta_anterior]==self.coincidencias[self.cuenta]):
387.                self.repite_paciente+=1
388.            else:
389.                self.repite_paciente=0
```

```
390.            if self.cambia_path==True:
391.                self.cuenta=0
392.                self.cambia_path=False
393.            if self.cuenta > self.max_cuenta:
394.                print("No hay mas imagenes")
395.            else:
396.                if(self.cuenta == 0):
397.                    input_image =
cv2.cvtColor(cv2.imread(os.path.join(self.data_path,self.images[self.cuenta]),1),cv2.COLOR_BGR2RGB)
398.                    self.cuenta +=1
399.                else:
400.                    self.cuenta+=1
401.                input_image =
cv2.cvtColor(cv2.imread(os.path.join(self.data_path,self.images[self.cuenta]),1),cv2.COLOR_BGR2RGB)
402.                for a in range(3,10):
403.                    if(self.ancho >= input_image.shape[1] / a):
404.                        alto2 = round(input_image.shape[0] /a)
405.                        ancho2= round(input_image.shape[1] /a)
406.                        break
407.                self.cv_img = Image.fromarray(cv2.resize(input_image,(ancho2,alto2)))
408.                print(round(input_image.shape[1]/ancho2)/round(input_image.shape[0]/alto2))
409.                #print(str(self.images[self.cuenta][0:self.long_ids[self.cuenta]]))
410.                self.root.photo=ImageTk.PhotoImage(self.cv_img)
411.                self.cv1.create_image(0,0, anchor = 'nw' , image=self.root.photo)
412.                self.T.delete('1.0',tk.END)
          #se vacia el cuadro de texto y se rellena con la siguiente info.
413.                self.T.insert(tk.END,str(self.images[self.cuenta]))
414.                self.cv1.bind("<Button-1>", self.callback)
415.                if(self.repite_paciente == 0):
416.                    self.cuentaevototal=0
417.                    self.cuentabiopsia=0
418.                    self.cuentaevo=0
419.                    self.textoHojaEvolucion(self.images[self.cuenta]
[0:self.long_ids[self.cuenta]])
420.                    self.textoHojaBiopsias(self.images[self.cuenta]
[0:self.long_ids[self.cuenta]])
421.
422.                self.update_idletasks()
423.            self.alto2 = alto2
424.            self.ancho2= ancho2
425.            self.T3.delete('1.0',tk.END)
       #se vacia el cuadro de texto y se rellena con la siguiente info.
426.            self.T3.insert(tk.END,str(self.cuenta)+'/'+str(self.max_cuenta))
427.
428.        def previaImagen(self):
429.            if self.cuenta == 1:
430.                print("No hay mas imagenes")
431.            else:
432.                self.cuenta -=1
433.                input_image =
cv2.cvtColor(cv2.imread(os.path.join(self.data_path,self.images[self.cuenta]),1),cv2.COLOR_BGR2RGB)
434.                for a in range(3,10):
435.                    if(self.ancho >= input_image.shape[1] / a):
436.                        alto2 = round(input_image.shape[0] /a)
437.                        ancho2= round(input_image.shape[1] /a)
438.                        break
439.                self.cv_img = Image.fromarray(cv2.resize(input_image,(ancho2,alto2)))
440.                print(str(self.images[self.cuenta][0:self.long_ids[self.cuenta]]))
441.                self.root.photo=ImageTk.PhotoImage(self.cv_img)
```

```
442.                self.cv1.create_image(0,0, anchor = 'nw' , image=self.root.photo)
443.                self.T.delete('1.0',tk.END)
          #se vacia el cuadro de texto y se rellena con la siguiente info.
444.                self.T.insert(tk.END,str(self.images[self.cuenta]))
445.                self.cv1.bind("<Button-1>", self.callback)
446.                if(self.repite_paciente == 0):
447.                    self.cuentaevototal=0
448.                    self.cuentaevo=0
449.                    self.cuentabiopsia=0
450.                    self.textoHojaEvolucion(self.images[self.cuenta]
[0:self.long_ids[self.cuenta]])
451.                    self.textoHojaBiopsias(self.images[self.cuenta]
[0:self.long_ids[self.cuenta]])
452.
453.                self.update_idletasks()
454.            self.T3.delete('1.0',tk.END)
       #se vacia el cuadro de texto y se rellena con la siguiente info.
455.            self.T3.insert(tk.END,str(self.cuenta)+'/'+str(self.max_cuenta))
456.
457.
458.        def textoHojaEvolucion(self,dato):
459.            self.cuentaevototal = 0
460.            self.bandera   = 0
461.            print(dato)
462.            self.evolucion_paciente =[]
463.            for i in range(self.total2[0]):
464.                if(self.evolucion[i][0]==dato):
465.                    print("llegahoja")
466.                    self.evolucion_paciente.append(self.evolucion[i])
467.                    self.bandera=1
468.                    self.cuentaevototal +=1
469.            if(self.bandera==0):
470.                cadena="No se tienen datos de evolucion de dicho paciente"
471.                self.T2.delete('1.0',tk.END)
472.                self.T2.insert(tk.END,cadena)
473.            else:
474.                self.T2.delete('1.0',tk.END)
475.                self.T2.insert(tk.END,self.evolucion_paciente[0])
476.            #print(self.evolucion_paciente)
477.
478.
479.        def textoSiguiente(self):
480.            #print(self.cuentaevo, self.cuentaevototal, self.evolucion_paciente)
481.            if(self.cuentaevo < self.cuentaevototal -1 and self.cuentaevototal != 0 ):
482.                self.T2.delete('1.0',tk.END)

483.                self.cuentaevo +=1
484.                print(self.cuentaevo)
485.                self.T2.insert(tk.END,str(self.evolucion_paciente[self.cuentaevo]))
486.            elif(not self.evolucion_paciente):
487.                cadena="No se tienen datos de evolucion de dicho paciente"
488.                self.T2.delete('1.0',tk.END)
489.                self.T2.insert(tk.END,cadena)
490.
491.
492.        def textoAnterior(self):
493.            print(self.cuentaevo)
494.            if(self.cuentaevo  > 0 and self.cuentaevototal > 0 and self.cuentaevototal != 0):
495.                self.T2.delete('1.0',tk.END)
```

```
496.                self.cuentaevo -=1
497.                self.T2.insert(tk.END,str(self.evolucion_paciente[self.cuentaevo]))
498.            elif(not self.evolucion_paciente):
499.                cadena="No se tienen datos de evolucion de dicho paciente"
500.                self.T2.delete('1.0',tk.END)
501.                self.T2.insert(tk.END,cadena)
502.
503.        def textoHojaBiopsias(self,dato):
504.            self.cuentabiopsiatotal = 0
505.            self.bandera2   = 0
506.            print(dato)
507.            del self.biopsiaPaciente[:]
508.            for i in range(self.total3):
509.                if(self.biopsias[i][0]==dato):
510.                    print("llegahojabiopsia")
511.                    self.biopsiaPaciente.append(self.biopsias[i])
512.                    self.bandera2=1
513.                    self.cuentabiopsiatotal +=1
514.            if(self.bandera2==0):
515.                cadena="No se tienen datos de evolucion de dicho paciente"
516.                self.T4.delete('1.0',tk.END)
517.                self.T4.insert(tk.END,cadena)
518.            else:
519.                self.T4.delete('1.0',tk.END)
520.                self.T4.insert(tk.END,self.biopsiaPaciente[0])
521.            #print(np.sort(self.biopsiaPaciente,axis=1))
522.
523.        def textoSiguiente2(self):
524.            print(self.cuentabiopsia, self.cuentaevototal)
525.            if(self.cuentabiopsia < self.cuentabiopsiatotal-1):
526.                self.T4.delete('1.0',tk.END)
527.                self.cuentabiopsia +=1
528.                print(self.cuentabiopsia)
529.                self.T4.insert(tk.END,str(self.biopsiaPaciente[self.cuentabiopsia]))
530.
531.
532.        def textoAnterior2(self):
533.            print(self.cuentabiopsia)
534.            if(self.cuentabiopsia  > 0):
535.                self.T4.delete('1.0',tk.END)
536.                self.cuentabiopsia -=1
537.                self.T4.insert(tk.END,str(self.biopsiaPaciente[self.cuentabiopsia]))
538.
539.
540.
541.        def guardaCrop(self):
542.            print(self.cuenta)
543.            print(self.images[self.cuenta])
544.            img=self.cropeada
545.            images=self.images
546.            contador=self.cuenta
547.            cuenta_anterior=self.cuenta_anterior
548.            repite_paciente=self.repite_paciente
549.            imagen_anterior=images[contador]
550.            guardaSinSobreescribir = False
```

```
551.            #print(type(contador))
552.            longids=self.long_ids
553.            longids=longids[contador]
554.            #print(self.out_path)
555.            #print(images[12][0:self.long_ids[12]])
556.            imgOrigRecorte = cv2.imread(os.path.join(self.data_path,str(images[contador])))
557.            RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
558.            if(self.coincidencias[cuenta_anterior]==self.coincidencias[contador]):
559.                repite_paciente+=1
560.                cuenta_anterior=contador
561.                print('Valor Repite_paciente: ',repite_paciente)
562.            else:
563.                repite_paciente=0
564.                cuenta_anterior=contador
565.                print('otro paciente')
566.                self.evolucion_paciente = 0
567.
568.            if(repite_paciente == 0 or self.cuenta == 0):
569.                cv2.imwrite(os.path.join(self.out_path,str(images[contador])),imgOrigRecorte)
#esta sera la nueva imagen que se guarda (una por cada imagen de la que se extraiga un recorte)
570.
571.            if(self.var1.get()==5):
572.                cv2.imwrite(os.path.join(self.out_path2,str(images[contador][0:longids])+'_'
            #caso especial en el que no se añade info. del contraste (se usara poco?)
573.                +str(self.patrones[self.var_boton.get()])+'_'+str(repite_paciente)
+'.jpg'),RGB_img)
574.            else:
575.                for m in range(len(self.recortesTotal)):
576.                    nombreImagen=str(images[contador][0:longids])
+'_'+str(self.cadenaNombreOctantes)+str(self.cadenaNombrePatrones)+'_'+str(m)+'.jpg'
577.                    #print("N imagen :"+str(nombreImagen)+" N recorte
guardado :"+str(self.recortesTotal[m]))
578.                    for k in range(m,len(self.recortesTotal)):
579.                        nombreImagenSiguiente = str(images[contador][0:longids])
+'_'+str(self.cadenaNombreOctantes)+str(self.cadenaNombrePatrones)+'_'+str(m+1)+'.jpg'
580.                        #print("N imagen: "+str(nombreImagen)+" N recorte
guardado :"+str(self.recortesTotal[k]))
581.                        #if(nombreImagen == str(self.recortesTotal[k])):
582.                            #nombreImagen=str(images[contador][0:longids])
+'_'+str(self.contrast[self.var1.get()])+'_'+str(self.patrones[self.var_boton.get()])+'_'+str(k)
+'.jpg'
583.                            #print(str(self.recortesTotal[k]))
584.                            #print(comprobacion[0])
585.                        print(k,len(self.recortesTotal))
586.                        if(k<len(self.recortesTotal)-1):
587.                            print(self.recortesTotal[k+1])
588.                            if(nombreImagen == str(self.recortesTotal[k]) and
(nombreImagenSiguiente != str(self.recortesTotal[k+1]))):
589.                                print("N imagen: "+str(nombreImagenSiguiente))
590.                                #print(nombreImagenSiguiente,self.recortesTotal[k+1])
591.                                #repite_paciente +=1
592.                                cv2.imwrite(os.path.join(self.out_path2,str(images[contador]
[0:longids])+'_'
593.                                +str(self.cadenaNombreBiospsias)+str(self.cadenaNombreOctantes)
+str(self.cadenaNombrePatrones)+'_'+str(m+1)+'.jpg'),RGB_img)
594.                                #print("llega")
595.                                guardaSinSobreescribir = True
596.                                break
597.                        else:
598.                            if(nombreImagen == str(self.recortesTotal[k])):
599.                                cv2.imwrite(os.path.join(self.out_path2,str(images[contador]
```

```
[0:longids])+'_'
                                +str(self.cadenaNombreBiospsias)+str(self.cadenaNombreOctantes)
+str(self.cadenaNombrePatrones)+'_'+str(m+1)+'.jpg'),RGB_img)
                                #print("llega")
                                guardaSinSobreescribir = True
                                break
                            if(nombreImagen != str(self.recortesTotal[k])):
                                cv2.imwrite(os.path.join(self.out_path2,str(images[contador]
[0:longids])+'_'
                                +str(self.cadenaNombreBiospsias)+str(self.cadenaNombreOctantes)
+str(self.cadenaNombrePatrones)+'_'+str(repite_paciente)+'.jpg'),RGB_img)
                                #print("llega")
                                guardaSinSobreescribir = True
                                break

                    if(guardaSinSobreescribir == True): break

            #si sale del bucle: no hay otra imagen de ese paciente
            if(guardaSinSobreescribir == False and repite_paciente==0):
                cv2.imwrite(os.path.join(self.out_path2,str(images[contador][0:longids])+'_'
                            +str(self.contrast[self.var1.get()])
+'_'+str(self.cadenaNombreOctantes)+str(self.cadenaNombrePatrones)+str(repite_paciente)
+'.jpg'),RGB_img)
            self.repite_paciente=repite_paciente
            self.cuenta_anterior=cuenta_anterior

self.coordenadas.append([self.posx,self.posy,self.tam,self.hei,self.wid,self.chan,self.cadenaNombreB
iopsias,self.images[self.cuenta],str(self.cadenaNombreOctantes),str(self.cadenaNombrePatrones)])
            self.recortesTotal = os.listdir(self.out_path2)
        #volvemos a leer los recortes  almacenados tras guardar uno
            with open("coordenadas.txt", "a") as myfile:
                myfile.write(str(self.coordenadas)+str('\n'))
                del self.coordenadas[:]
            #Se añade la posibilidad de que se guarde una copia de la imagen en otra carpeta para
distinguir aquellas de las que se extrae un recorte

    if __name__ == "__main__":
        root = tk.Tk()
        app=imageCropper(root)
        tk.mainloop()
```